НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

Кафедра автоматизованих систем обробки інформації і управління

До захисту допущено:
В.о. завідувача кафедри
_____ *Олександр ПАВЛОВ*
(підпис) (вл.ім'я, прізвище)
"___"_____2020 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інформаційні управляючі системи та технології»
спеціальності 122 «Комп'ютерні науки та інформаційні технології»

на тему: «**Клієнт-серверна Web-система для роботи з фінансами**»

Виконав:	студент *IV* курсу, групи *ІС-61*	
	Касьян Євгеній Володимирович (прізвище, ім'я, по батькові)	_____ (підпис)
Керівник	*доц., к.т.н., Жураковська Оксана Сергіївна* (посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові)	_____ (підпис)
Консультант з графічної документації	*доц., к.т.н., доц. Тєлишева Тамара Олексіївна* (посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові)	_____ (підпис)
Рецензент	*Доцент кафедри ТК ФІОТ, к.т.н., доцент Лісовиченко Олег Іванович* (посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові)	_____ (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет (інститут) *інформатики та обчислювальної техніки*
(повна назва)

Кафедра *автоматизованих систем обробки інформації та управління*
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 122 «Комп'ютерні науки та інформаційні технології»

Освітньо-професійна програма «Інформаційні управляючі системи та технології»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ *Олександр ПАВЛОВ*
(підпис) (вл.ім’я, прізвище)
“___”_____2020 р.

З А В Д А Н Н Я
на дипломний проєкт студенту

Касьяну Євгенію Володимировичу
(прізвище, ім’я, по батькові)

1. Тема проєкту «Клієнт-серверна Web-система для роботи з фінансами» керівник проєкту *Жураковська Оксана Сергіївна к.т.н.*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від “*7*”*травня* 2020 р. №1081-с
2. Термін подання студентом проєкту “*01*”*червня* 2020 року
3. Вихідні дані до проєкту
Технічне завдання

4. Зміст пояснювальної записки
1. Загальні положення: основні визначення та терміни, опис предметного середовища, огляд ринку програмних продуктів, постановка задачі
2. Інформаційне забезпечення: вхідні дані, вихідні дані, опис структури бази даних
3. Математичне забезпечення: змістовна та математична постановки задачі, обґрунтування та опис методу розв’язання
4. Програмне та технічне забезпечення: засоби розробки, вимоги до технічного забезпечення, архітектура програмного забезпечення, побудова звітів
5. Технологічний розділ: керівництво користувача, методика випробувань програмного продукту
5. Перелік графічного матеріалу
1. Схема структурна класів програмного забезпечення

2. Схема структурна послідовності

3. Схема структурна компонентів програмного забезпечення

4. Креслення вигляду екранних форм

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання *«13» квітня 2020 року*

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	*Вивчення рекомендованої літератури*	*13.04.2020*	
2.	*Аналіз існуючих методів розв'язання задачі*	*15.04.2020*	
3.	*Постановка та формалізація задачі*	*17.04.2020*	
4.	*Розробка інформаційного забезпечення*	*24.04.2020*	
5.	*Алгоритмізація задачі*	*28.04.2020*	
6.	*Обґрунтування використовуваних технічних засобів*	*30.04.2020*	
7.	*Розробка програмного забезпечення*	*05.05.2020*	
8.	*Налагодження програми*	*06.05.2020*	
9.	*Виконання графічних документів*	*08.05.2020*	
10.	*Оформлення пояснювальної записки*	*10.05.2020*	
11.	*Подання ДП на попередній захист*	*15.05.2020*	
12.	*Подання ДП на основний захист*	*01.06.2020*	
13.	*Подання ДП рецензенту*	*02.06.2020*	

Студент Євгеній КАСЬЯН

Керівник Оксана ЖУРАКОВСЬКА

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	ДП 6111.00.000 ПЗ	Пояснювальна записка	76	
3	А4	ДП 6111.01.000 ТЗ	Технічне завдання	10	
4	А3	ДП 6111.02.000 ССК	Схема структурна класів програмного	1	
			забезпечення		
5	А3	ДП 6111.03.000 ССП	Схема структурна послідовності	1	
6	А3	ДП 6111.04.000 ССК	Схема структурна компонентів	1	
			програмного забезпечення		
7	А3	ДП 6111.05.000 КЕ	Креслення вигляду екранних форм	2	

Зм.	Арк.	ПІБ	Підп.	Дата	ДП 6111 00.000.00		
Розробн.		Касьян Є.В.			Відомість дипломного проєкту	Літ.	Лист 1 / Листів 1
Керівн.		Жураковська О.С.					
Консульт.						КПІ ім. Ігоря Сікорського Каф. АСОІУ Гр. ІС-61	
Н/контр.		Тєлишева Т.О					
В.о.зав.каф.		Павлов О.А.					

Пояснювальна записка
до дипломного проєкту

на тему: Клієнт-серверна Web-система для роботи з фінансами

Київ – 2020 року

АНОТАЦІЯ

Структура та обсяг роботи. Пояснювальна записка дипломного проєкту складається з 5 розділів, містить 19 рисунків, 13 таблиць, 1 додаток, 23 джерела.

Дипломний проєкт присвячений розробці комплексу задач фінансового контролю для підвищення простоти моніторингу особистих фінансів та коштів. Задачею системи є спрощення процесу документації особистих фінансових транзакції (будь то прибуток, витрата, або переказ на свій інший рахунок).

У розділі інформаційного забезпечення були визначенні вхідні дані для усіх функціональних варіантів використання програми: реєстрація користувача, створення транзакції декількома варіантами, створення планування бюджету для користувача. Також було описано вихідні дані та вигляд у якому вони будуть додаватися до колекцій бази даних. Окрім цього у розділі було описано структуру бази даних.

Розділ математичного забезпечення присвячений опису математичних операцій, які було впроваджені до проекту для реалізації усіх поставлених задач розробки.

Розділ програмного забезпечення описує всі принципи роботи всіх засобів розробки, які були використані у дипломному проєкті, та обґрунтування чому обрана кожна за них. Також у цьому розділі описано архітектурну частину програмного забезпечення, та описано специфікацію розроблених функцій.

У технологічному розділі описана інструкція для використання програмного забезпечення, а також проведено тестування розробленої системи.

ВЕБ-РОЗРОБКА, ФІНАНСОВИЙ МЕНЕДЖМЕНТ, КОНТРОЛЬ ГРОШОВОГО ПОТОКУ, ФІНАНСОВИИЙ ТРЕКЕР, ОСОБИСТИЙ КАПІТАЛ, АНАЛІТИКА, FRONT-END, BACK-END.

ABSTRACT

Structure and scope of work. The explanatory note of the diploma project consists of 5 sections, contains 19 drawings, 13 tables, 1 application, 23 sources.

The diploma project is devoted to the development of a set of financial control tasks to increase the ease of monitoring personal finances and funds. The goal of the system is to simplify the process of documenting personal financial transactions (whether it is profit, expense, or transfer to your other account.

In the section of information support the input data for all functional variants of use of the program were defined: registration of the user, creation of transaction by several variants, creation of budget planning for the user. The source data and the form in which they will be added to the database collections were also described. In addition, the section described the structure of the database and described its visual representation.

The section of mathematical support is devoted to the description of mathematical operations which have been introduced to the project for realization of all set tasks of development.

The software section describes all the principles of operation of all development tools that were used in the thesis project, and the rationale for choosing each of them. This section also describes the architectural part of the software, and describes the specification of the developed functions.

The technological section describes the instructions for using the software, as well as testing the developed system.

WEB DEVELOPMENT, FINANCIAL MANAGEMENT, CASH FLOW CONTROL, FINANCIAL TRACKER, PERSONAL CAPITAL, ANALYTICS, FRONT-END, BACK-END.

ЗМІСТ

ВСТУП

У нас час люди кожен день втрачають свої гроші, тому постає необхідність контролювати особистий фінансовий потік. Управління особистим грошима стосується індивідуального багатства та доходу, покриття фінансових потреб, та досягнення поставлених цілей.

Кожен з нас займається роботою більшість свого життя, та отримує за це дохід. Кожна людина хоче ефективно витрачати те, що вона заробляє, але не завжди хватає часу на контролювання витрат та прибутків. Тому постає питання контролю особистих фінансів.

Це питання не є новим. З часу початку капіталізму, економічна політика сильно змінилася, та контроль фінансів, став грати важливу роль для кожної фінансово грамотної людини.

Незалежно від того ким ви працюєте, ви точно маєте основні витрати щомісяця, Це зрозуміло, але ще є багато інших статей витрат на які уходить також багато грошей. В ідеалі ви повинні заробляти достатньо грошей щомісяця, щоб закрити всі ваші потреби та мати заощадження, щоб відкладати їх на майбутнє. Але, щоб заробляти достатньо, та покривати всі потреби, як правило, зарплатні недостатньо, тому постає задача з створення свого особистого капіталу. Але для цього потрібно виявити, та проаналізувати усі компоненти фінансового планування:

- оцінка;
- цілі;
- розробка плану;
- виконання;
- моніторинг та переоцінка.

Багато людей витрачають всі свої кошти, а інколи навіть більше ніж заробляють, тому через деякий час вони опиняються у борговій ямі. Звичайно ж це все негативно впливає на економіку країни. Тому мені здається що треба популяризувати опанування фінансової грамотності. Основою фінансової

грамотності є контролювання грошового потоку, та відкладання частини прибутку. Також одним із важливих та потужних інструментів — є планування особистого бюджету. Це дозволяє створити план витрат, для розподілу особистий грошей таким чином, щоб їх хватало на все: покриття своїх потреб, та відкладання для формування капіталу.

Ваш бюджет, є основним інструментом для досягнення успіху. Він дає вам можливість контролювати вашу фінансову майбутнє, та будувати фінансову незалежність.

Облік можна вести, використовуючи звичайний листок паперу, або ексель-табличку, але оскільки зараз століття цифрових технологій, та час спливає дуже швидко, немає можливості вести облік, присвячуючи цьому багато свого часу, чи мати прив’язку до місця. Тому було вирішено розробити систему, яка дозволяє оптимізувати цей процес та вести облік, за допомогою будь-якого пристрою, який має з’єднання з інтернетом.

Тому дипломний проєкт присвячений розробці системи, які дозволяють планувати бюджет та моніторити прибутки та витрати.

1 ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1 Опис предметного середовища

В наш час постає проблема витрати своїх коштів. І людина у системі сучасного миру витрачає свої кошти декілька раз на день. Інколи буває так, що через деякий час неможливо зрозуміти, куди були витрачені кошти.

Для вирішення цієї задачі було вирішено розробити систему, за допомогою якої, користувач зможе одразу фіксувати свої трати, та подалі аналізувати та змінювати статті своїх витрат.

1.1.1 Опис процесу діяльності

Кожна людина веде контроль своїх фінансів і зустрічається за проблемою системи контролю.

Діяльність контролю коштів передбачає записи доходів та витрат, вирахування на які категорії скільки витрачається коштів, де можна скоротити витрати, а куди навпаки: можна направити фінанси. Тому ця система спрямована на автоматизацію ведення обліку доходів та витрат. Тобто система допомагає простіше вести облік та економити час на обчислення всіх математичних операцій які постають у задачі ведення фінансів, та спростити управління своїми коштами.

Окрім цього система стає доступною з будь-якого простою, який має доступ до інтернету, що також є більш пріоритетним способом контролю, ніж звичайний «листок бумаги», або Excel.

1.1.2 Опис функціональної моделі

Основною дійовою особою програмного продукту буде користувач.

Користувач буде мати такі можливості:

- реєстрація / авторизації;
- додавання фінансового рахунку;

- додавання транзакції (витрата прибуток переказ);
- перегляд зроблених транзакцій;

Модель варіантів використання зображено на Рис. 1.1

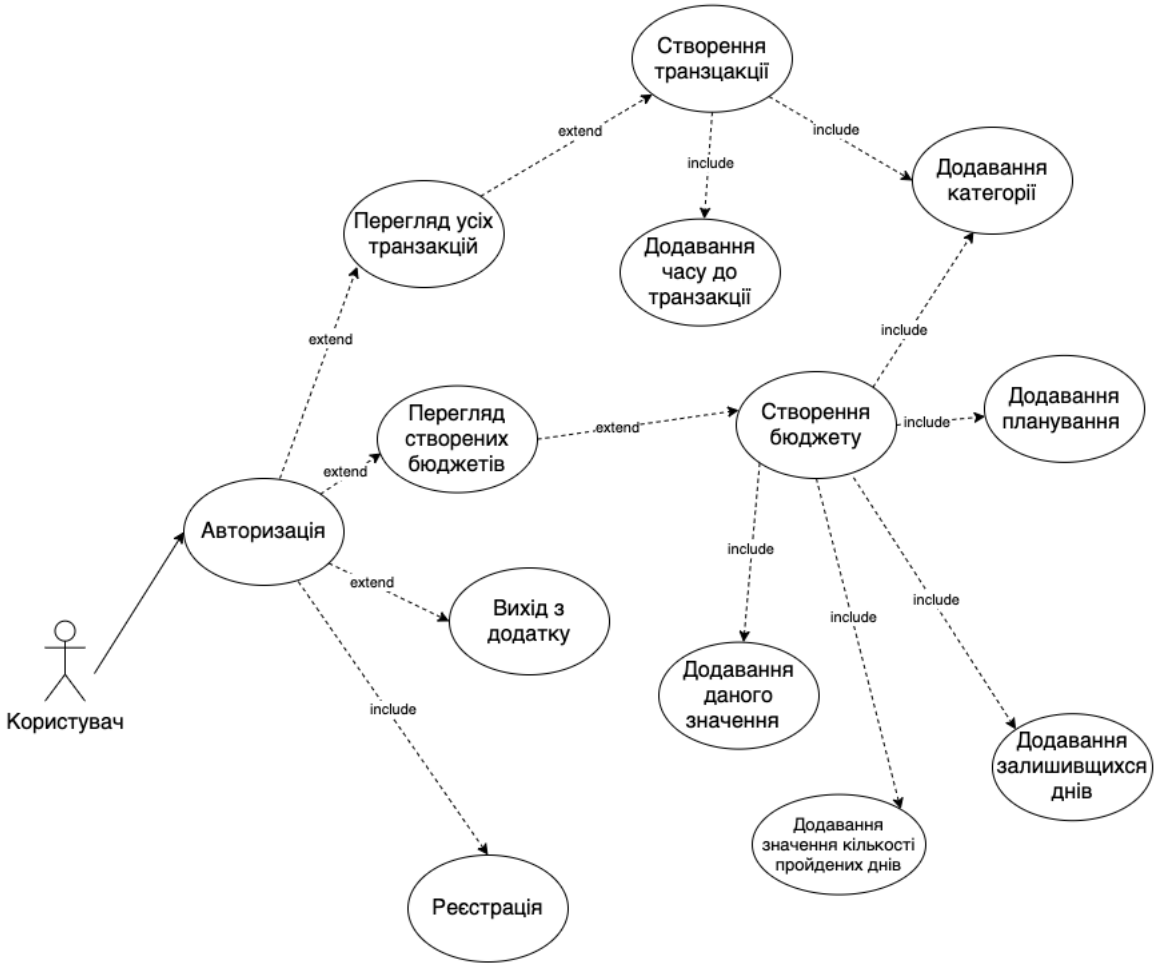


Рисунок 1.1 — Схема структурна варіантів використань

Функціональні вимоги:

а) обов’язково повинна бути система реєстрації та авторизації;

б) система повинна робити розрахунки на основі нових транзакцій (доходів, витрат, переказів);

в) вести базу транзакцій та мати можливість змінювати данні.

1.2 Огляд наявних аналогів

На сьогодні на ринку існую декілька аналогічних продуктів, але кожен з них має кілька недоліків:

1.2.1 Wallet від budgetbakers

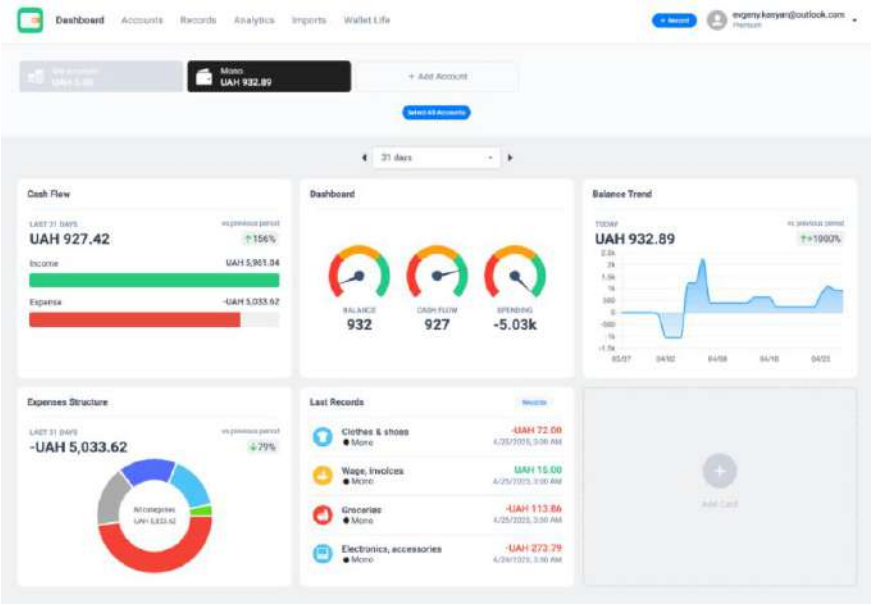


Рисунок 1.2 – Програма-аналог Wallet

Переваги:

а) зображення Інформації на графіках;

б) розбиття транзакцій на категорії;

в) можливість додати декілька Рахунків.

Недоліки:

а) складний і інколи незрозумілий інтерфейс для користувача;

б) висока вартість підписки на користування програмою.

1.2.2 Програма-аналог ZenMoney

Рисунок 1.3 – Програма-аналог ZenMoney

Переваги:

- можливість створювати план бюджету та цілей;
- можливість додати декілька рахунків.

Недоліки:

- застарілий, та часто незрозумілий інтерфейс;
- часто не правильно працює програма.

1.2.3 Cash Wallet Spendee

Рисунок 1.4 – Програма-аналог Cash Wallet Spendee

Переваги:

- зображення Інформації на графіках;
- розбиття транзакцій на категорії;
- можливість додати декілька Рахунків.

Недоліки

- складний і інколи незрозумілий інтерфейс для користувача;
- недостатньо інформативний інтерфейс;
- застарілі концепції дизайну.

Висновок: в наслідок незручності інших застосунків, вони не є актуальними. Тому було вирішено розробити зручний і легкий у використанні сервіс, щоб популяризувати серед користувачів даний тип додатку з ціллю менеджменту грошового потоку.

1.3 Постановка задачі

1.3.1 Призначення розробки

Призначенням розробки є підтримка процесів контролю фінансових операцій та планування бюджету.

1.3.2 Цілі та задачі розробки

Основною метою розробки є спрощення процедури контролю особистих фінансів, забезпечення доступності цієї процедури, а також спрощення та підвищення ефективності процедури планування бюджету.

Для цього потрібно вирішити такі задачі:

- реєстрація користувачів;
- авторизація користувачів
- створення журналу фінансових операцій;
- планування бюджету на обраний період;
- додавання категорій за якими будуть здійснюватися операції;
- формування звітів.

Висновок до розділу

У даному розділі було детально описано предмети середовище, короткий опис управління фінансами, опис процесу діяльності. А також було розписано, навіщо потрібно така система.

Далі була описана функціональна модель, яка відповідає за варіанти використання системи, створені функціональні вимоги, та описані можливості програми які потрібно реалізувати.

Також у цьому розділі були проаналізований аналогічний програми та виявленні їх недоліки.

Результатом цього розділу є огляд призначення розробки та сформовані цілі та задачі розробки.

2 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Вхідні дані

Для даного застосунку вхідні дані можуть бути декілька типів, а саме вхідні дані для транзакції та для бюджету, тому далі наведено перелік вхідних даних для кожного з них.

При додаванні транзакції (ручне введення даних) вхідними є такі дані:

- тип транзакції (прибуток / витрата / перерахунок);
- розмір транзакції;
- рахунок для якого виконується транзакція;
- категорія транзакції;
- дата та час транзакції;
- ідентифікатор транзакції (генерується автоматично).

Далі на рисунку 2.1 можна побачити приклад об’єкту з набором даних

```
{
  type: "income",//   Тип транзакції (прибуток / витрата / перерахунок)
  ammount: 15000,//   Розмір транзакії в мінімальних одиницях валюти
  account: "kKGVoZuHWzqVoZuH",//   Рахунок для якого виконується транзакція
  category: "taxi",//  Категорія транзакції
  date: 1552392228,//  Дата та час транзакції
  id: "ZuHWzqkKGVo=",//  Ідентифікатор транзакції (генерується автоматично)
}
```

Рисунок 2.1 — Візуальне представлення даних при ручному вводі транзакції

При додаванні транзакції автоматично за допомогою API банківського сервісу:

- унікальний id транзакції;
- час транзакції в секундах в форматі Unix time;
- опис транзакції;

- код типу транзакції;
- статус блокування суми;
- сума у валюті рахунку в мінімальних одиницях валюти (копійках, центах);
- код валюти рахунку;
- розмір комісії в мінімальних одиницях валюти;
- розмір кешбеку в мінімальних одиницях валюти;
- баланс рахунку в мінімальних одиницях валюти.

На рисунку 2.2 можна побачити приклад об’єкту з цим набором даних

```
{
"id": "ZuHWzqkKGVo=",    //Унікальний id транзакції
"time": 1554466347,    //Час транзакції в секундах в форматі Unix time
"description": "Покупка щастя",    //Опис транзакцій
"mcc": 7997,    //Код типу транзакції (Merchant Category Code), відповідно ISO 18245
"hold": false,    //Статус блокування суми (детальніше у wiki)
"amount": -95000,    //Сума у валюті рахунку в мінімальних одиницях валюти (копійках, центах)
"currencyCode": 980,    //Код валюти рахунку
"commissionRate": 0,    //Розмір комісії в мінімальних одиницях валюти (копійках, центах)
"cashbackAmount": 19000,    //Розмір кешбеку в мінімальних одиницях валюти (копійках, центах)
"balance": 10050000    //Баланс рахунку в мінімальних одиницях валюти (копійках, центах)
}
```

Рисунок 2.2 — Візуальне представлення даних при автоматичному запиті транзакції

При створенні планування бюджету, потрібний інший набор вхідних даних:

а) категорія для якої створюється планування бюджету;

б) значення бюджету, який планується;

в) значення, яке показує, скільки вже витрачено з даного бюджету;

г) період у днях, скільки залишилось, до завершення бюджету;

д) період на котрий плануються зробити планування бюджету у днях.

```
{
  category: "Home", // Категорія для якої створюється планування бюджету
  plan: 4000, //   Значення бюджету, який планується
  current: 3023, //  Значення, яке показує, скільки вже витрачено з даного бюджету
  daysLeft: 7, //  Період у днях, скільки залишилось, до завершення бюджету
  daysTotal: 20 //   Період на котрий плануються зробити планування бюджету у днях
}
```

Рисунок 2.3 — Візуальне представлення даних для планування бюджету

2.2 Вихідні дані

Результатом роботи системи інформаційного застосунку є наступний перелік вихідних даних(сигналів):

а) оброблена інформація записується у відповідні таблиці бази даних;

б) результат обробки вхідної інформації до системи записується до документу який бачить користувач(Html документ);

в) результат функції котра розраховує залишок на рахунку;

г) результат функції котра вираховує витрати або прибутки за категоріями.

Після обробки вхідних даних, вихідні дані записується у відповідну колекцію бази даних, сутність яких наведено у таблиці 2.1

Таблиця 2.1 — Таблиці бази даних з відповідними даними

Колекція	Дані	Значення полю
Transaction	Id	Унікальний id транзакції
	time	Час транзакції в секундах
	description	Опис транзакцій

Продовження таблиці 2.1

Колекція	Дані	Значення полю
	mcc	Код типу транзакції
	hold	Статус блокування суми
	amount	Сума у валюті рахунку в мінімальних одиницях валюти (копійках, центах)
	currencyCode	Код валюти рахунку
	commissionRate	Розмір комісії в мінімальних одиницях валюти (копійках, центах)
	cashbackAmount	Розмір кешбеку в мінімальних одиницях валюти (копійках, центах)
	balance	Баланс рахунку в мінімальних одиницях валюти (копійках, центах)
Users	Id	Унікальний id користувача
	firsName	Ім’я користувача
	lastName	Прізвище користувача
	Email	Email користувача
	Password	Пароль користувача
	phoneNumber	Телефон користувача
	userName	Логін користувача

Продовження таблиці 2.1

Колекція	Дані	Значення полю
Accounts	urerId	id користувача
	id	Id рахунку
	Amount	Баланс
	currencyCode	Код валюти рахунку
budget	urerId	id користувача
	category	категорія для якої створюється планування бюджету;
	plan	значення бюджету, який планується;
	current	значення, яке показує, скільки вже витрачено з даного бюджету;
	daysLeft	період у днях, скільки залишилось, до завершення бюджету;
	daysTotal	період на котрий плануються зробити планування бюджету у днях.

Перелік вихідних документів:

а) візуальне представлення даних на сторінці у вигляді списку транзакцій;

б) баланс рахунку.

2.3 Опис структури бази даних

Для даного програмного забезпечення була реалізована база даних MongoDB яка складається з 4 колекцій:

- Users;
- Budget;

- Accounts;
- Transactions.

Кожна колекція відповідає за свій набір даних, а саме:

- Users, відповідає за зберігання користувачів сервісу;
- Budget, відповідає за зберігання бюджетів користувачів;
- Accounts, відповідає за рахунки користувачів;
- Transactions, відповідає за транзакції рахунків.

Далі на схемі бази даних, рисунок 2.4 можна побачити як кожна з колекції з'єднана з іншою

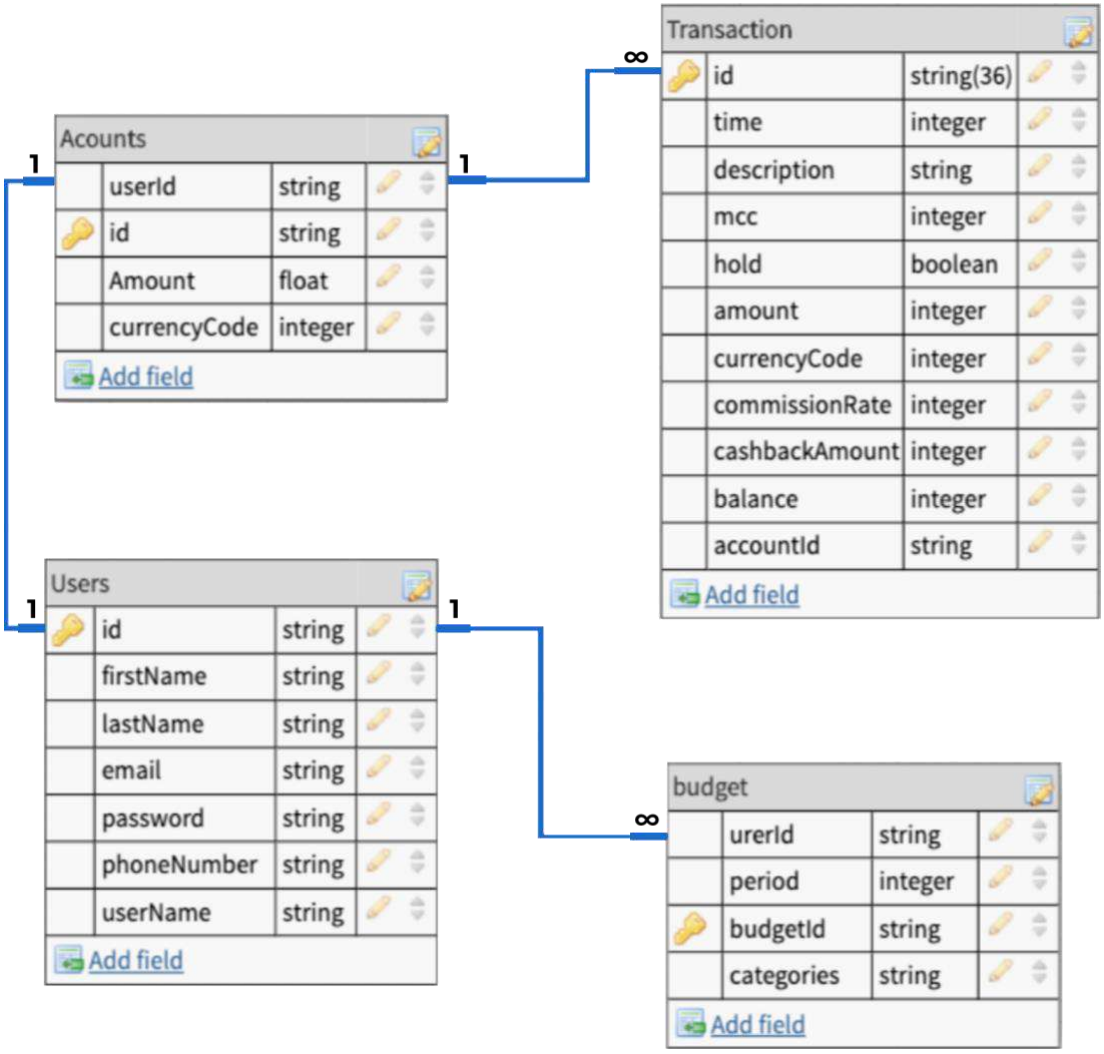


Рисунок 2.4 – Схема структурна бази даних

Висновок до розділу

У цьому розділі було описано інформаційне забезпечення, яке відповідає за опис вхідних даних та вихідних даних, структуру бази даних, та запроваджено її до проєкту. Також розглянені вхідні та вихідні дані та їх приклади.

Також наведено опис кожної колекції та характеристики її полей.

У вихідних даних було описано вигляд то якому вони будуть додаватися до колекцій бази даних.

3 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Постановка задачі

Оскільки дипломна робота, спрямована на спрощення процесу персонального фінансового менеджменту, то у цьому розділі буде розглянуто математичні задачі, використання яких дозволяє вирішувати грошові проблеми, та обчислювати час для реалізації поставленої цілі.

У цьому розділі, розглянемо виршення таких задач, як:

- амортизація;
- обрахування «простого відсотка»;
- обрахування «складного відсотка»;
- обрахування грошового потоку;
- оцінка річного темпу зростяння;

3.2 Обґрунтування методів розв’язання

В свою чергу задача планування бюджету зводиться до математичної задачі аналізу витрат для подальшого вкладання фінансів у інструменти інвестування, тому для планування вкладу грошей, потрібно мати чітке уявлення про ці інструменти, та правильно обрахувати та проаналізувати результати задач. Усі ці задачі, активно використовують інформацію про конкретні області фінансового менеджменту, тому це дозволяє ефективно зробити аналіз інвестування.

3.3 Опис методів розв’язання

Розглянемо кожну задачу окремо.

Задача амортизації. Задача амортизації означає здійснення періодичних платежів за деякий час для погашення боргу. Рішення цієї задачі спрямовано на розрахування, розміру щомісячного платежу. Вирішення цієї задачі

потрібно для розрахунку вартості довгострокової заборгованості, наприклад, іпотеки, кредиту на авто, тощо. Амортизація обраховується за формулою (3.1).

$$A = \frac{P*\left(\frac{r}{n}\right)}{(1-\left(1+\frac{r}{n}\right)^{-n*t}}, \quad (3.1)$$

де А - амортизація;

Р - сума займу;

r - вартість позичання;

n – строк займу.

Наступна задача обрахування «простого відсотка» Простий відсоток - це відсоток, отриманий від основної суми. Цей розрахунок можна зробити швидко, щоб дати уявлення про те, скільки відсотків буде нараховуватися за час. Вирішення цієї задачі потрібно для оцінки того, скільки вийде заробити на депозитному рахунку, або скільки потрібно заплатити за кредіт. Цей показник обраховується за формулою (3.2).

$$A = P * r * t, \quad (3.2)$$

де А – вихід;

Р – сума займу або депозиту;

r – вартість позичання;

t – час займу, або депозиту.

Задача обрахування складного відсотка. Складні відсотки - це відсотки, отримані від основної суми, та будь-які відсотки, нараховані в минулому. Використовується замість простого рівняння відсотків, щоб отримати більш точне значення, відсотків: котрі будуть нараховані. Потрібен для обрахування фактичного відсотку, який можна заробити за час інвестицій або потрібно сплатити за борг. Обраховуються за формулою (3.3).

$$A = P(1 + \frac{r}{n})^{nt}, \quad (3.3)$$

де А – вихід;

Р – початковий внесок;

r – річна відсоткова ставка;

n – кількість разів складання відсотку за рік;

t – кількість років.

Задача обрахування грошового потоку. Грошовий потік показує, скільки ви заробляєте по відношенню до того, скільки ви витрачаєте. Дозволяє подивіться, чи живете ви в межах своїх можливостей. Якщо число від'ємне, ви витрачаєте занадто багато, то це означає, що Ви витрачаєте більше, ніж заробляєте. Обраховується за формулою (3.4).

$$F = I - E, \tag{3.4}$$

де F – грошовий потік;

I – загальний прибуток;

E –загальні витрати.

Задача оцінки річного темпу зростяння. Потрібен, щоб визначити темп приросту інвестицій протягом декількох років. Визначте середній темп приросту акцій, облігацій, портфеля, нерухомості чи будь-якого типу інвестицій протягом кількох років. Розраховується за формулою (3.5).

$$CAGR = (\frac{E}{S})^{(\frac{1}{t})} - 1, \tag{3.5}$$

де CAGR – сукупний середньорічний темп зростання

E – початкове значення параметра;

S – кінцеве значення параметра;

t – кількість років.

Висновок до розділу

В даному розділі була сформульовані математичні задачі, та описані методи використання яких дозволяє вирішувати грошові проблеми, та обчислювати час для реалізації поставленої цілі. У розділ описано кожен з цих методів, та приведе обґрунтування до них, чому вони є ефективними. Також наведено та описано формули для вирішення кожної з задачі.

4 ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

4.1 Засоби розробки

Під час розробки програмного забезпечення було використано програмно-технічні технології для розробки клієнтської частини, серверної частини та проектування бази даних.

Технології котрій були використані для клієнтської частини:

- мова програмування JavaScript;
- бібліотека React;
- бібліотека Redux;
- препроцесор Sass.

Для серверної частини:

- Платформа Node js;
- середовище для обробки запитів на сервері.

Для зберігання даних було використано нереляційну базу MongoDB

Увесь програмний продукт розроблявся у таких програмах, як

- Visual Studio Code - програмний редактор коду, розроблений та оптимізований для проектування, тестування та впровадження сучасних веб-додатків;
- Postman – застосунок для спрощення процесу посилання запитів та створення кращого API;
- Браузер Chrome та Safari – web-браузери для перегляду сторінок у інтернеті;
- Командний рядок - спеціальна програма, яка дозволяє управляти комп'ютером шляхом введення текстових команд з клавіатури;
- Sketch - інструментарій для дизайну, для проектування, та створення інтерфейсів та UI-UX дизайну;
- Zeplin інструменту для дизайну та проектування веб, iOS, Android та React Native інтерфейсів, та спрощення їх візуалізації;

- Javascript – функціональна мова програмування для застосування складних речей на веб-сторінці, та динамічного додавання, імплементування та обробки вводимої та виводимої інфомації ;

  На сьогодення на javscript написана багато бібліотек та фреймворків, та це єдина широко підтримуємая мова для роботи клієнтської частини веб – застосунку.

Найпопулярніші бібліотеки написані на javascript:

- jQuery: Швидка, маленька і багатофункціональна бібліотека JavaScript, широко використовується для спрощення роботи з DOM, Html та для розширення базового функціоналу [1];
- BackBoneJS: надає структуру веб-додаткам, надаючи моделі з прив’язкою до ключових значень та спеціальними подіями, колекції з багатим API численних функцій, перегляди з декларативним обробкою подій та з'єднує все це з вашим існуючим API через інтерфейс REST-ful JSON [2];
- D3.js: Бібліотека JavaScript для управління документами на основі даних та візуалізації. Ця бібліотека допоможе вам реалізувати Ваші дані у графічному вигляді, за допомогою HTML, CSS та SVG [3];
- React: ефективна, швидка, декларативна бібліотека JavaScript від Facebook, розроблена для швидкого та спрощеного створення веб-додатків та користувацьких інтерфейсів [4];
- Underscore.js: це набір функцій-утиліт, до яких звикли любителі функціонального програмування, Ruby, Python або Prototype.js (але, на відміну від Prototype ця бібліотека не має права впроваджувати базові класи JavaScript). Вона була написана, щоб добре уживатися з jQuery [5];
- Moment.js: це відмінна бібліотека для роботи з датами в JavaScript, допомагає швидко відформатовати дати, та перевести їх до потрібного вигляду, використовуючи лише виклик потрібної функції [6];

- Lodash: це бібліотека, з набором корисних функцій, для роботи з даними, для конвертації їх з одного формату в інший, фільтрації і інших речей [7].

Front-end Frameworks зазвичай складаються з пакету, який містить у собі файли та папки, такі як HTML, CSS, JavasScript і т.д. Також існує багата кількість автономних фреймворків:

- Bootstrap: HTML, CSS і JS фреймворк для швидкої розробки інтерактивних та гнучких інтерфейсів для десктопних та мобільних проектів [8];
- Foundation: це потужний CSS-фреймворк, який є альтернативою для Bootstrap [9];
- Semantic UI: це фреймворк для створення переносимих інтерфейсів, який допоможе повторно використовувати елементи UI в своїх проектах [10];
- uikit: це легка, модульна платформа (фреймворк) для розробки швидких і потужних веб-інтерфейсів [11].

Web Application Frameworks – каркас для створення динамічних та гнучких веб-застосунків, мережевих додатків, сервісів або ресурсів. Він спрощує розробку і позбавляє від необхідності написання рутинного коду.

- Ruby - середовище розробки, яка чудово підходить для створення будь-якого типу веб-додатків: систем для управління веб-сайтами і платформ для ведення електронної торгівлі, програм для організації спільної роботи і для веб-сервісів для здійснення комунікації, для облікових і ERP-систем , статистичних та аналітичних систем [12];
- Angular - представляє фреймворк від компанії Google для створення клієнтських додатків. Перш за все він націлений на розробку SPA-рішень (Single Page Application), тобто односторінкових додатків. В цьому плані Angular є спадкоємцем іншого фреймворка AngularJS [13];

- Ember.js - це продуктивна, перевірена на бій JavaScript рамка для створення сучасних веб-додатків. Він включає все, що потрібно для створення багатих інтерфейсів, які працюють на будь-якому пристрої [14];
- Express – швидкий фреймворк, який дозволяє зручно налаштувати роботу сервера на NodeJs [15].

Для створення проекту було використано бібліотеку React оскільки його функціоналу достатньо для реалізації клієнтської частини інтерфейсу та front-end частини проекту. Усіх його функціональних можливості вистачило для реалізації дипломних задач та поставлених проблем. Завдяки своїй функціональності, React дозволив декомпозувати проект на невеликі частини, які називаються компоненти.

Окрім бібліотеки, react проекті було використано такі модулі:

- Express - модуль для налаштування та обробки запитів на сервері, тобто написання серверной частини додатку [15];
- Nodemon - утиліта, яка контролює будь-які зміни у репозиторії та автоматично перезапускає локально створений сервер. Ідеально підходить в якості допоміжного інструменту для розробки серверної частини;
- node-sass бібліотека яка забезпечує прив’язку NodeJs до CSS стилей, та дозволяє швидко та при кожній зміні файлу стилей транспілювати файли SASS у CSS;
- create-react-app бібліотека, яка пропонує спосіб створення React додатку, вона пропонує сучасні методи та способи налаштування зборки без конфігурації;
- react-dom пакет, який забезпечує специфічні для DOM методи він застосований для з’єднання з універсальним пакетом React, котрий поставляється як реагуючи на npm;

- mongoose це бібліотека моделювання даних об’єктів (ODM) для MongoDB та Node.js. Він управляє зв'язками між даними, забезпечує перевірку схеми і використовується для перекладу між об'єктами в коді та представлення цих об'єктів у MongoDB;
- nodeJS – платформа яка базується на движку Google V8, Та дозволяє розроблювати серверні додатки. Його схема роботи зав’язана на транспортування програмного коду у машиний для подальшого використання [16];
- babel бібліотека з набором інструментів, котрий дозволяє використовувати перетворення коду в У зворотній сумісний код в інших браузерах або середовищах [17];
- css_linter потужний лінтер для знаходження помилок, котрій допомагає забезпечити виконання правил у стилях до документу;
- esLint аналізую ваш javascript код для того щоб знайти помилки та проблеми, дозволяє писати чистий, легко-читаємий та підтримуємий код.

Node.js представляє єдину платформу для написання JavaScript-програми, запропонованої із застосуванням зовнішніх бібліотек.

Завдяки Node.js, написаному для коду браузера, JavaScript отримує доступ до глобальних об'єктів, таким чином, як документ і вікно, а також з іншими API та бібліотеками. Це робить можливим написання абсолютно будь-яких застосунків: від використання командної строки та відеоігри до полноценних веб-серверів.

Найчастіше Node.js використовується при написанні веб-пропозицій з інтенсивним вводом-виводом. Самій розповсюджений приклад - це веб-сервери. Node.js використовує для створення запропонованого реального часу: чатів, комунікаційних програм та ігор. Багато додатків Node.js містить і серверну, і клієнтську частини.

Разом з NodeJs, також постачається пакетний менеджер npm. У ході багатьох років Node широко використовував розробки JavaScript для обміну інструментами, встановлення різних модулів та управління їх залежностями.

Npm - це широко використовуваний репозиторій для публікації проектів Node.js з відкритим ісходним кодом. Це означає, що це онлайн-платформа, де кожен може публікувати та декларувати свої інструменти, написані на JavaScript.

Також npm - це інструмент командної строки, який допомагає взаємодіяти з онлайн-платформами, такими як браузери та сервери. Ця програма використовує програму для встановлення та видалення пакетів, керування версіями та залежностями, необхідними для використання проекту.

Окрім всіх цих бібліотек та фреймворків, у проектуванні проекту використовувалась система контролю версій Git .

GIT дозволяє розробити можливість отримати багато абсолютно незалежних гілок коду. Створення, видалення та об’єднання цих гілок відбувається без будь-яких проблем і великого затрату часу.

В GIT всі операції атомарні - це означає, що будь-яка дія може бути вдалою, або провалитися. Це дійсно важливо, так як у деяких системах контролюю версій (CVS), де дії не є атомарними, деякі запрошені операції, можуть залишити його в нестабільному стані.

На відміну від інших VCS, таких як SVN або CVS, де метадані зберігаються в скритих папках (.cvs, .svn і т. Д.), В GIT всі дані розміщені в каталогах .git. Він використовує модель даних, яка допомагає забезпечити криптографічну цілісність всього, що присутня в репозиторіях. Кожен раз, коли файли додаються або коммітяться, генеруються їх контрольні суми; аналогічний процес відбувається через їхні вилучення.

Ще одна цікава функція, присутня в GIT - це його індекс. У попередніх індексах, розробники можуть сформулювати комміти та переглянути їх до фактичного застосування.

4.2 Вимоги до технічного забезпечення

Для повноцінної роботи розробленої програми, необхідним технічним засобом є комп'ютер з встановленим браузером, котрий підтримує javascript, та доступом до інтернету.

Мінімальними вимогами до компьютеру є:

- об'єм оперативної пам'яті 2gb;
- процесор з тактовою частотою 2ГГц.

4.2.1 Загальні вимоги

Для підтримки програмного забезпечення, компьютер, має бути підключеним до інтернету, та на ньому мають бути встановлені:

- редактор програмного коду;
- пакетний менеджер NPM.

4.3 Архітектура програмного забезпечення

Архітектура програмного забезпечення складається з 2 застосувань:

- клієнтська частина додатку для впровадження користувацького інтерфейсу, розроблена на основі бібліотеки React «client-structure-interface».

- WEB-API застосування «Back-End» для впровадження функціоналу для збереження, змінення та отримання даних з бази даних, для подальшої їх обробки та передачі до клієнтської частини проекту. Також це застосування відповідає за створення локального серверу для впровадження інтерфейсу та функціоналу обробки даних. В свою чергу, це застосування поділяється на 2 частини. Частина бізнес-логіки, яка відповідає за обробку запитів та передачу їх далі, а також за передачу відповіді на клієнт. Та частина, яка відповідає за впровадження бази даних.

Основними перевагами клієнт-серверної архітектури є розділення клієнтської та серверної частини на окремі незалежні частини. Така

архітектура дозволяє підключать «клієнта» до «серверу» у комп'ютерній мережі [18].

Серверна сторона цієї архітектури забезпечує центральну функціональність, тобто будь-яка кількість клієнтів може підключатися до одного серверу, та запросити виконання задачі, яка була відправлена з клієнтської частини. У свою чергу, сервер приймає ці запити, виконує необхідне завдання та повертає потрібні результати клієнту, якщо того потребує запит [18].

Для кращого розуміння цієї архітектури, розглянемо цю систему на прикладі онлайн-бібліотеки. Припустимо, що додаток дозволяє шукати у великій кількості книжок, потрібну, переглядати деталі кожної з них, та замовляти обрану книжку. Все, що бачить користувач, тобто увесь інтерфейс, для пошуку, для перегляду інформації, сама інформація, все це обробляється на клієнтській частині додатку. А от коли користувач обирає вподобану книгу, то клієнтська частина посилає запит на сервер, де ці данні оброблюються, та відправляються назад до клієнтської частини.

З іншої сторони, додаток може мати форму монолітної архітектури, завантаженої один раз. Тому, кожного разу, коли щось змінюється, або оновлюється, потрібно наново оновлювати весь додаток. Очевидно, що це потребує значно більше ресурсів. Якщо повернутися до прикладу, то можна сказати, що така «монолітна архітектура», не спрацювала би, оскільки додаток, оновлюється дуже часто. Окрім цього, якщо копія всього додатку буде зберігатися у кожного користувача, і буде відсутня серверна частина, додаток стає зовсім «небезпечним», оскільки у кожного, хто завантажив копію додатку, будуть користувацькі дані інших користувачів.

Удосконаленням є поділ програми на 2 частини: клієнтську та серверну. З першої клієнт може отримати інтерфейс для користування додатком. А інша частина зберігається та працює на власному серверному пристрої компанії.

Схему роботи простої клієнт-серверної архітектури зображено на рисунку 4.1. Це одночасно вирішує проблему безпеки, та проблему оновлення даних кожного разу при зміненні самого додатку.

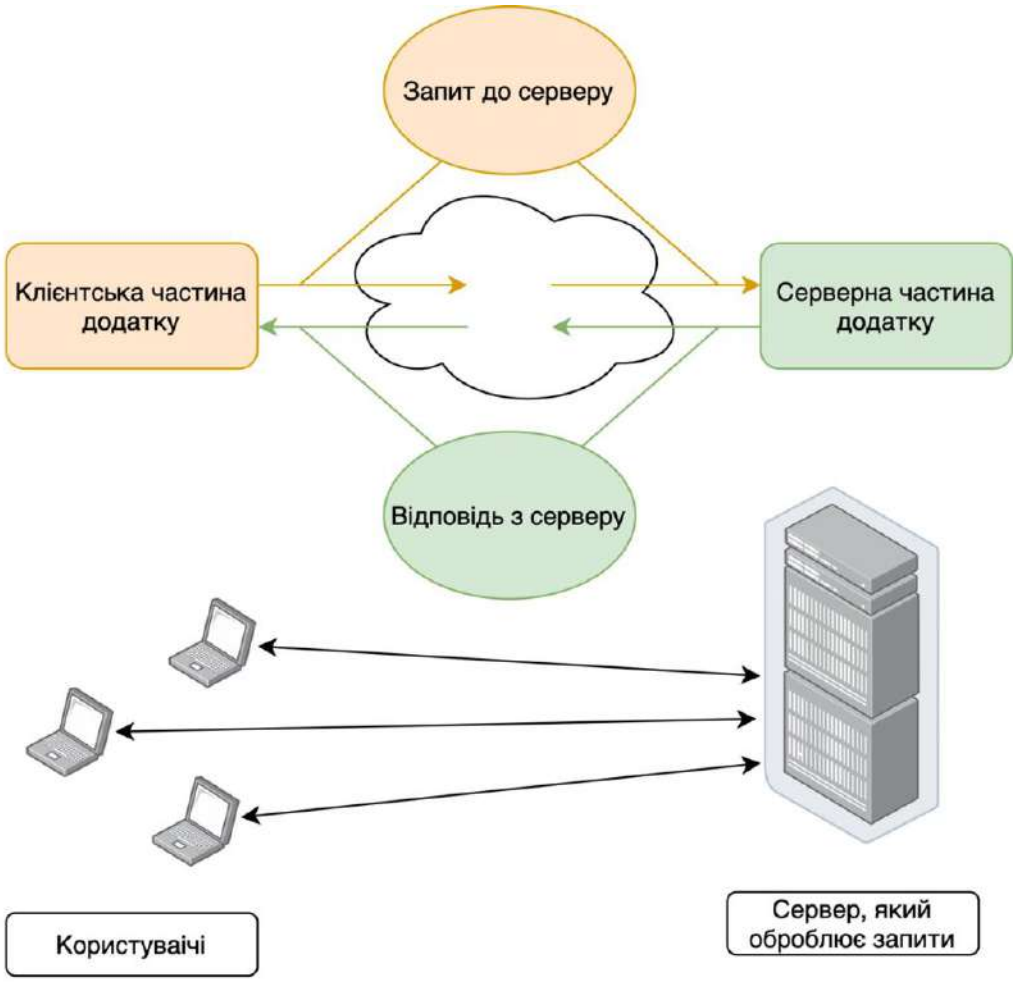


Рисунок 4.1 – Схема структурна клієнт-серверної дворівневої архітектури

На цій моделі є багато «клієнтів», котрі з’єднуються з сервером, для запиту потрібної інформації. Сервер обробляє цей запит і далі посилає відповідь до клієнту. Доки клієнт має з’єднання з серверною частиною, сервер буде оброблювати всі поступаючі запити з клієнту, так же як це би працювало використовуючи монолітну архітектуру.

Уся інформація яка потрібна для роботи додатку, централізовано зберігається на сервері, та за допомогою серверної частини додатку, через

змінення серверної частини, уся інформація може оновлюватися. Це дозволяє завжди мати актуальну інформацію для усіх клієнтів одночасно.

Оскільки ця архітектура не дозволяє впроваджувати підтримку доступу персональної інформації, тому у дипломному проекту, було використано трирівневу клієнт-серверну архітектуру, де третім рівнем є реалізація бази даних [18]. Ця схема забражена на рис 4.2

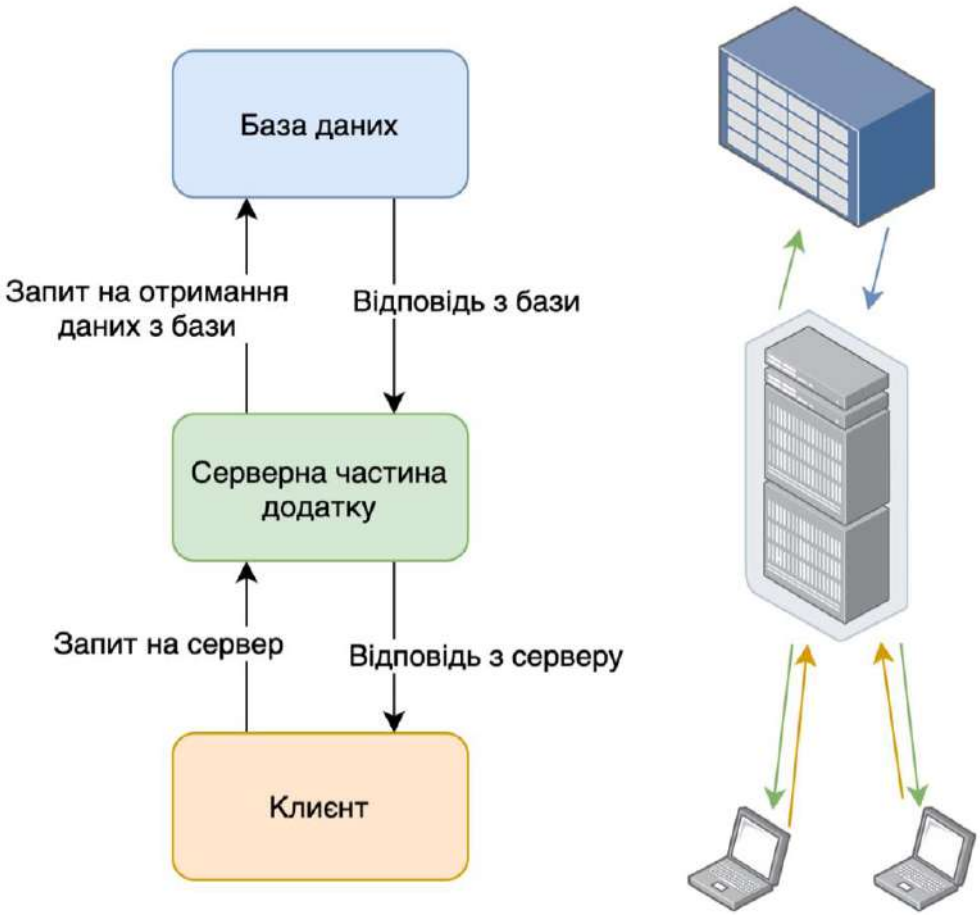


Рисунок 4.2 – Схема структурна клієнт-серверної трирівневой архітектури.

Третій рівень розширює реалізацію систем и за допомогою бази даних. Цей рівень відповідає за збереження даних користувачів. Розширення системи третім рівнем, має багато переваг: швидкість розробки, масштабованість, продуктивність і доступність. Окрім цього додається модульність, для зберігання даних у системі, ця модульність дозволяє розроблювати та покращувати продукт з більшою швидкістю, та мінімальними зміненнями

інших рівні додатку. Також це дозволяє розділяти ці слої між різними розробниками.

Якщо ви маєте розділені слої додатку, ви можете збільшити доступність, та надійність додатку. Також це дозволяє розміщувати різні частини вашого додатку на різних серверах. Також трирівнева архітектура запроваджую безпечність для усієї системи.

Для клієнтської частини було використано бібліотеку React та Redux, які імплементовані за допомогою MVC архітектури [19].

M – model – надає дані та методи роботи з цими даними, також реагує на запити, змінюючи свій стан. Ще модель не повинна містити інформацію, як ці данні можна візуалізовувати.

V – view – відповідає за візуальне представлення інформації.

C – controller – впроваджує зв’язок між користувачем та системою. Контролює введення даних користувачем, та дозволяє використовувати модель та її представлення для реалізації дії системи.

Web-Api застосунок, тобто серверна частина, розроблена на основі REST архітектури [20]. За допомогою цієї архітектури ми можемо посилати запити з клієнтської частини на серверну. REST архітектура дозволяє нам використовувати різні типи запитів до серверу, але у проєкті були використані такі запити:

- GET – запитує представлення вказаного ресурсу. Запити, які використовують GET, повинні лише отримувати дані [21];

- POST – використовується для відправки об'єкта на вказаний ресурс, але цей метод може часто викликати зміну стану або побічних ефектів на сервері [21];

- PUT – замінює всі поточні представлення цільового ресурсу на корисне навантаження, що вказане в запиті [21];

- DELETE – видаляє вказаний ресурс [21].

Рівень даних реалізовано за допомогою бази даних Mongo DB – нереляційної бази даних. Вона дозволяє управляти багато-вузловими, та складними кластерами, дуже просто та швидко, також є змога відслідковувати стани кластерів та інтегрувати їх з іншими рішеннями. MongoDB має високий рівень захисту, тому турбуватися про це не має сенсу.

4.3.1 Діаграма класів

Діаграма класів приведена у графічному матеріалі до дипломного проєкту.

4.3.2 Діаграма послідовності

Діаграма послідовності допомагає детально описати спосіб виконання операцій. Вона фіксує взаємодію між об'єктами та їх діями. Діаграма послідовності орієнтована на час і вона візуально показую порядок взаємодії, використовуючи вертикальна вісь діаграми, щоб поставити час, як повідомлення надсилається і коли [23].

Тому для більш простого, та наглядного представлення передачі повідомлень між компонентами у застосунку, було створено діаграму послідовності, на якій можна чітко зрозуміти як система реагує на дії користувача, та які процеси виконується.

Діаграма послідовності приведена у графічному матеріалі до дипломного проекту.

4.3.3 Діаграма компонентів

Діаграма компонентів розбиває розроблену систему на різні високі рівні функціональності. Кожен компонент відповідає за одну чітку мету у всій системі і лише взаємодіє з іншими істотними елементам, якщо це потрібно. Діаграма компонентів, також наведена у графічному матеріалі дипломного проекту.

4.3.4 Специфікація функцій

Розглянемо специфікацію розроблених та використаних функції у даному застосунку.

Таблиця 4.1 – Функції класів програмного забезпечення

Назва	Опис
Клієнтська частина застосунку «Registration.js»	
validateEmail (email)	Функція для перевірки поля email при реєстрації нового акаунту
validatePassword (pass)	Функція для валідації паролю при реєстрації нового акаунту.
vilidateSecondPassword (pass)	Функція для перевірки ідентичності паролю у першому полі та у другому при реєстрації нового акаунту.
Клієнтська частина застосунку «User_account.js»	
addTransaction (type, amount, account, category, date)	Функція, яка дозволяє додати транзакцію, до візуального представлення її на клієнтській частині, та викликає функцію, яка записую ці данні у базу даних. Ця функція приймає: - тип транзакції; - розмір транзакції; - рахунок для виконання транзакції; - категорію транзакції; - дату та час виконання транзакції

Продовження таблиці 4.1

Назва	Опис
getTime (date)	Конвертує отримуємую дату до формату, котрий потрібен для візуалізації
Клієнтська частина застосунку «Budget_account.js»	
addBudget (category, period, amount)	Створення нового планування бюджету по категорії, на період, та на введенне значеня, які приймає функція.
changeBudget (category, period, newAmount)	Функція оновлює бюджет, працює так само, як і функція створення бюджету
deleteBudget (category)	Ця функція дозволяє видалити обраний бюджет, та повертаю інформацію, про успішність виконання, або ні.

Треба зазначити, що для серверної частини було реалізовано три-шарову структуру, для реалізації бізнес-логіки. Тому кожен шар відповідає за відповідну логіку. Про кожен за них та їх функції, все описано в таблиці 4.2

Таблиця 4.2 – реалізація архітектурної частини додатку, та її функції

Перший шар: services - частина серверної реалізації, яка відповідає, за реалізацію бізнес-логіки. У проекті використано 3 таких класу, кожен з яких відповідає за користувача, транзакції, та бюджет відповідно.	
Назва	Опис
userService.js – відповідає за бізнес логіку користувача	

Продовження таблиці 4.2

Назва	Опис
createUser(data)	Функція, яка отримує об’єкт з даними користувача, та створює, його, якщо користувача не існує. Якщо користувач вже існую у базі, то користувач отримує відповідне повідомлення
updateUser(data)	Функція, яка отримує об’єкт з даними користувача, то оновлю, його, якщо користувача не існує. Якщо користувач вже існую у базі, то користувач отримує відповідне повідомлення
deleteUser(login)	Функція, яка отримує логін користувача, та видаляє з бази цього користувача.
search(string)	Функція, яка отримує строку, та перевіряє, існує такий користувач у базі, чи ні, у разі відсутності, цього користувача, повертає відповідне повідомлення.
transactionService.js	
addTransacrion(data)	Функція, яка отримує об’єкт з даними про транзакцію, та додає її у базу.
removeTransaction(id)	Функція, яка отримує ідентифікатор транзакції, та видаляє її з бази.

Продовження таблиці 4.2

Назва	Опис
editTransaction(data	Функція, яка отримує об’єкт з даними про транзакцію, та оновлює її у базі.
budgetService.js	
addBudget(data)	Функція, яка отримує об’єкт з даними про створений бюджет, та додає її у базу.
removeBudget(id)	Функція, яка отримує ідентифікатор бюджету, та видаляє його з бази.
editBudget(data)	Функція, яка отримує об’єкт з даними про бюджет, та оновлює його у базі.
Наступний слой – repositories – слой, який відповідає, за збереження даних, та він має свої функції далі буде розглянуті всі використані функції для користувача (user), але подібні функції також реалізовані для бюджету та транзауції.	
generateId	Генерує унікальний ідентифікатор
getUser(search)	Приймає об’єкт по даним якого здійснюється пошук у існуючій базі
createUser(data)	Функція записує створеного користувача до бази даних, та повертає булеве значення, того успішно це було зроблено, чи ні.
createUser(data)	Функція перезаписує оновленого користувача до бази даних, та повертає булеве значення, того успішно це було зроблено, чи ні.

Продовження таблиці 4.2

Назва	Опис
deleteUser(id)	Видаляє обраного користувача з бази даних.
Аналогічно до попередніх двох шарів, цей шар, має актуальність для усіх оброблюємих даних. Цей шар називається middleware, та відповідає він за проміжні обробку поступаючих даних, саме тут відбувається первірка, та валідація даних, котрі поступають на сервер, а далі записуються до бази даних.	

Висновок до розділу

У розділі програмного технічного забезпечення, було описано засоби розробки за допомогою яких був зроблений проект, було обґрунтовано вибір цих технологій, та наведені їх переваги.

Також у цьому розділі були описані вимоги до технічного забезпечення, та загальні вимоги.

У підрозділи архітектура програмного забезпечення, було описано у варіанти, за якими можна створювати веб додаток, розглянуто декілька варіантів архітектури продукту, та обгрунтовано, чому обрана саме трирівневой архітектура. Вона включає в себе рівень клієнтської частини додатку, серверної частини додатку та рівень бази даних.

Також у цьому розділі було наведено опис 3 діаграм: діаграма компонентів, діаграму послідовності, діаграма класів. До кожної з цих діаграм, було створено графічний матеріал, який знаходиться у розділі дипломного проекту «графічний матеріал».

Далі у цьому розділі було наведено специфікацію функції. Були розглянуті найважливіші функції які використані у застосунку. Опис всіх функцій був поділений на декілька частин. Функції клієнтської частини — тут були описані функції, які відповідають за клієнтську частину, та виконують

найважливіші операції з компонентами. Та функції серверної частини, оскільки серверна частина була також розбита на 3 рівні, то було описано переваги цього розбиття, та описано за що відповідає кожен з рівнів і відповідно описані функції до кожного з рівнів.

5 ТЕХНОЛОГІЧНИЙ РОЗДІЛ

5.1 Керівництво користувача

Для початкового роботи з проєктом, користувач повинен, завантажити всі додаткові пакети через пакетний менеджер yarn або npm. Далі потрібно виконати запуск локального серверу, для цього потрібно перейти у репозиторій з проектом, та у командній столиці написати команду «npm run start». Після цього проект збирається та запускається на локальному сервері. Далі потрібно відкрити браузер і перейти на локальний сервер “http://localhost:3000”.

Після цього користувач побачить головну сторінку додатку (рисунок 5.1)

Рисунок 5.1 – Головна сторінка програми

Розглянемо детальніше домашню сторінку користувача. На цій сторінці ми бачимо 2 кнопки: «Log in» та «Sing in», 1-а та 2-а відповідно. 1-а кнопка відповідає за авторизацію користувача у системі, якщо він вже зареєстрований. Кнопка номер 2, відповідає за реєстрацію користувача у системі для подальшого її використання. Якщо користувач натиснув на кнопку 1, тоді відкривається сторінка для авторизації (рисунок 5.2). А якщо натиснув кнопку 2, то відкривається сторінка реєстрації (рисунок 5.3).

Рисунок 5.2 – Сторінка реєстрації

На цій сторінці наявна форма для авторизації. Під цифрою 1 ми бачимо поле для вводу логіну або email. Під цифрою 2 позначено поле, для вводу паролю користувача. Кнопка 3 дозволяє авторизуватися користувачу через Facebook. Кнопка 4 дозволяє авторизуватися користувачу через Google. Кнопка 5 — відправляє введені дані на сервер, та перевіряє наявність користувача у базі даних і відповідно відповідає назад на клієнтську частину.

Рисунок 5.3 – Сторінка реєстрації

На сторінці реєстрації є 4 поля для ведення інформації:

- поле для введення email, яке валідується коли користувач робить його неактивним, також перевіряється наявність такого адресу у базі;
- поле логін обов’язково для вводу та перевіряється чи нема дублікатів у базі;
- поле password також обов’язково для ведення та повинно мати мінімально довжину 6 символів та включати в себе і букви і цифри;
- поле Re-enter Password повинно повторювати поле password.

Коли користувач авторзується, то в нього відкривається головна сторінка особистого кабінету (рисунок 5.4).

Рис 5.4 – Головна сторінка персонального кабінету

На головній сторінці персонального кабінету ми бачимо багато функціональних елементів.

- за допомогою кнопки годин, ми можемо додати фото користувача;
- у полі 2 ми можемо змінити ім’я та прізвище користувача;
- список під номером 3 відповідає за список транзакцій котрі будуть додані користувачем;
- бокове меню відповідає за управлінням сторінками програми, тобто можна переключатися зі сторінки транзакцій до сторінки бюджет;
- під номером 5 ви можете бачити кількість грошей на вашому рахунку;
- Функціональний елемент під номером 6, відповідає за додавання транзакції, розглянемо його детальніше (рисунок 5.5).

Рисунок 5.5 – Форма для додавання транзакції

Ця форма потрібно щоб створити нову транзакцію. За допомогою кнопок 1, 2, 3 на рисунку 5.5, користувач обирає тип транзакції, тобто прибуток, витрата або переказ. Далі за допомогою поля 4, користувач водить розмір транзакції. У поле категорія під номером 5, користувач записує категорію до якої відноситься транзакція. У поле під номером 6 користувач записує час, коли була виконана транзакція. Після того як всі поля заповнені, потрібно надіслати форму на сервер, для цього створена кнопка під номером 7.

На рисунку 5.6, представлено, такий вигляд набуває особистий кабінет у розділі транзакції коли створені транзакції, додана фотокартка та обрахований залишок на рахунку.

Рисунок 5.6 – заповнений особистий кабінет

Оскільки програма також підтримує планування бюджету, далі на рисунку 5.7, можна побачити цю сторінку. На цій сторінці є декілька полей для вводу інформації та кнопка додати. Ці поля відповідають за категорію, запланований бюджет, поточні витрати, кількість залишившихся днів до закінчення цього планування, та загальна кількість днів.

Рисунок 5.7 – сторінка планування бюджету

Як можна побачити далі на рисунку 5.8, планування бюджету відбувається таким чином, що кожна нова категорія, додається новою строкою. Якщо загальні витрати за категорію перевищують заплановані витрати за категорією, то загальні витрати позначаються червоним кольором. Кожна з цих рядків зберігається у базі даних, тому нема потреби хвилюватися що ваші дані кудись зникнуть.

Рисунок 5.8 – Сторінка зі сформованим бюджетом

Також в ході розробки програмного забезпечення виникла необхідність, у створенні сторінки, яка би з'являлася якщо той запит який хоче отримати користувач не існує. Тому було вирішено зробити ще одну сторінку з помилкою 404. Ця сторінка зображено далі на рисунку 5.9.

Рисунок 5.9 – Незнайдена сторінка

Якщо користувач потрапляє на цю сторінку, у нього є можливість повернутися на домашню сторінку за допомогою кнопки «Go to Homepage»

5.2 Випробування програмного продукту

В цьому підрозділі описано тести та порядок тестування готового продукту для порівняння відповідності програмного забезпечення до задач, які були поставлені функціональним вимогам, розписаними у технічному завданні.

5.2.1 Мета випробувань

Метою випробувань - є перевірка відповідності функцій комплексу задач контролю особистих фінансів, тобто моніторинг транзакцій, прибутків, витрат та вимогам технічного завдання.

5.2.2 Загальні положення

Випробування проводяться на основі наступних документів:

– ГОСТ 34.603–92. Інформаційна технологія. Види випробувань автоматизованих систем;
– ГОСТ РД 50-34.698-90. Автоматизовані системи вимог до змісту документів.

5.2.3 Результати випробувань

Кожна частина програмного продукту була протестована відповідність функціональним вимогам, також була перевірена роботоздатність програми, на правильність.

Нижче розглянуто результати та зміст тестування. Тести підпорядковують у собі тестування роботи серверної частини, тобто валідація форм, перевірка наявності інформації в базі даних, також тестування проведено і для клієнтської частини, а саме для перевірки роботоспроможності форм, навігації додатку, проведено тестові сценарії додавання транзакції та бюджету у таблицях 5.1 - 5.7.

Таблиця 5.1 — Тестування працездатності кнопки авторизація на відкриття форми авторизації

Тест	Кнопка авторизації при правильній роботі відкриває сторінку з формою авторизації
Початковий стан системи	Відкрита головна сторінка з кнопками авторизації та реєстрації
Дія	Натискаємо на кнопку авторизації
Очікуваний результат	Відкриється сторінка з формою авторизації
Фактичний результат співпадає з очікуваним	Так

Таблиця 5.2 — тестування працездатності кнопки реєстрації на відкриття форми реєстрації

Тест	Кнопка реєстрації при правильній роботі відкриває сторінку з формою реєстрації
Початковий стан системи	Відкрита головна сторінка з кнопками авторизації та реєстрації
Дія	Натискаємо на кнопку реєстрації
Очікуваний результат	Відкриється сторінка з формою реєстрації
Фактичний результат співпадає з очікуваним	Так

Таблиця 5.3 — тестування працездатності форми реєстрації на відкриття форми реєстрації

Тест	Форма реєстрації при правильній роботі відкриває сторінку користувача, та додає користувача до бази даних
Початковий стан системи	Відкрита сторінка з формою реєстрації
Дія	Заповнюємо всі поля у відповідному форматі
Очікуваний результат	Відкриється головна сторінка особистого кабінету.
Фактичний результат співпадає з очікуваним	Так

Таблиця 5.4 — тестування працездатності форми авторизації на відкриття форми реєстрації

Тест	Форма авторизації при правильній роботі відкриває сторінку користувача
Початковий стан системи	Відкрита сторінка з формою авторизації
Дія	Заповнюємо всі потрібні поля у відповідному форматі
Очікуваний результат	Якщо введено правильний логін та пароль, відкриється головна сторінка особистого кабінету.
Фактичний результат співпадає з очікуваним	Так

Таблиця 5.5 — тестування працездатності функціоналу для додавання зображення у особистий кабінет

Тест	При натисканні на кнопку + у верхньому лівому краї форми, користувач може обрати фотографію яка буде виступати у ролі аватарки
Початковий стан системи	Відкрита головна сторінка акаунта користувача
Дія	Додаємо зображення
Очікуваний результат	Зображення збережено у базі даних та відображається на сайті
Фактичний результат співпадає з очікуваним	Так

Таблиця 5.6 — тестування меню авторизованого користувача

Тест	В лівому меню користувача повинні бути посилання на інші сторінці та вони мають працювати
Початковий стан системи	Відкрито особистий кабінет користувача
Дія	Почергово нажимаємо на посилання у боковому меню
Очікуваний результат	Сторінка переключається на ту за яке відповідає обране посилання
Фактичний результат співпадає з очікуваним	Так

Таблиця 5.7 — тестування функціоналу додавання транзакції авторизованого користувача

Тест	Форма додавання транзакції створює нову транзакцію
Початковий стан системи	Користувач авторизований та відкрита сторінка транзакцій
Дія	Користувач заповню форму додавання транзакції, обирає тип транзакції, заповнюємо розмір транзакції, обирає категорія та дату, після цього натискаємо кнопку створити
Очікуваний результат	нова транзакція додається у елемент останні транзакції
Фактичний результат співпадає з очікуваним	Так

Таблиця 5.8 — тестування функціоналу додавання бюджету для авторизованого користувача

Тест	Форма давання транзакції створює нову транзакцію
Початковий стан системи	Користувач авторизований та відкрита сторінка бюджету
Дія	Користувач заповнює форму додавання бюджету, після цього натискаємо кнопку «+»
Очікуваний результат	Новий бюджет додається до списку бюджетів
Фактичний результат співпадає з очікуваним	Так

Таблиця 5.9 — тестування валідації email для неавторизованого користувача

Тест	Перевірка валідації email, для користувача, котрий хоче зареєструватися
Початковий стан системи	Система знаходиться на сторінці реєстрації
Дія	Користувач заповнює у формі реєстрації поле, яке відповідає за email.
Очікуваний результат	Коли email введено невірно, система нотифікує користувача про це, якщо email введено вірно, система дає можливість відправити дані на сервер
Фактичний результат співпадає з очікуваним	Так

Таблиця 5.10 — тестування валідації паролю для незареєстрованого користувача

Тест	Перевірка валідації паролю, для користувача, котрий хоче зареєструватися
Початковий стан системи	Система знаходиться на сторінці реєстрації
Дія	Користувач заповнює у формі реєстрації поле, яке відповідає за пароль.
Очікуваний результат	Якщо пароль не виконує умови валідації, система нотифікує корстувача про це, якщо все добре, система досзволяє перейти до наступного кроку
Фактичний результат співпадає з очікуваним	Так

Таблиця 5.11 — тестування Функціонал показу неіснуючої сторінки

Тест	Якщо сторінка у формі не існує, система переходить на створену сторінку «404»
Початковий стан системи	Будь-який
Дія	Користувач переходить на сторінку, якої не існує в системі
Очікуваний результат	Відображається сторінка «404»
Фактичний результат співпадає з очікуваним	Так

Висновок до розділу

У цьому розділі було розроблено керівництво користувача, яке включає в себе опис інтерфейсу, опис всіх функціональних елементів, опис всіх можливостей системи. Для кожної сторінки була розроблена інструкція по використанню, розписано як користувач може зареєструватися, авторизуватися, додати фотокартку особистому кабінеті, додати транзакцію, створити персональний бюджет. Також були створені скріншоти роботи програми, які наглядно показують як виглядає система.

У розділі випробовування програмного продукту наведено мету та описані загальні положення випробувань, також були проведені самі випробування. Описана кожне з проведених випробувань, які дії були виконані, протестовані. Було перевірено, чи відповідають очікувані результати, фактичним.

ЗАГАЛЬНІ ВИСНОВКИ

У пояснювальній записці детально розглянуто опис дипломного проєкту, котрий присвячен спрощенню контролю особистих фінансів.

Детально описані предметне середовище, приведено короткий опис управління фінансами та опис процесу діяльності. А також було розписано потрібність такої системи. Була описана функціональна модель, яка відповідає за варіанти використання системи, створені функціональні вимоги, та описані можливості програми які потрібно реалізувати. У першому розділі були проаналізований аналогічний програми та виявленні їх недоліки. Проведено огляд та описане призначення розробки, сформовані цілі та задачі розробки.

У розділі опису інформаційного забезпечення, описано вхідні та вихідних дані, розроблена структура бази даних, та запроваджено її до проєкту. Також розглянені вхідні та вихідні дані та їх приклади.

У розділі програмного технічного забезпечення, було описано засоби розробки за допомогою яких був зроблений проект, було обґрунтовано вибір цих технологій, та наведені їх переваги.

Також були описані вимоги до технічного забезпечення, та загальні вимоги.

У підрозділи архітектура програмного забезпечення, було описано варіанти, за якими можна створювати веб додаток, розглянуто декілька варіантів архітектури продукту, та обґрунтовано, чому обрана саме трирівнева архітектура. Тут же було наведено опис 3 діаграм: діаграма компонентів, діаграму послідовності, діаграма класів. До кожної з цих діаграм, було створено графічну схему, який знаходиться у розділі дипломного проекту «графічний матеріал».

Далі було наведено специфікацію функцій. Були розглянуті найважливіші функції які використані у застосунку. Опис всіх функцій був поділений на декілька частин.

У останньому розділі роботи було розроблено керівництво користувача, яке включає в себе опис інтерфейсу, опис всіх функціональних елементів, опис всіх можливостей системи. Для кожної сторінки була розроблена інструкція по використанню, розписано як користувач може зареєструватися, авторизуватися, додати фотокартку особистому кабінеті, додати транзакцію, створити персональний бюджет.

У підрозділі випробовування програмного продукту наведено мету та описані загальні положення випробувань, також були проведені самі випробування. Описано кожне з проведених випробувань, які дії були виконані та протестовані. Проаналізовано, чи відповідає фактичний результат, очікуваному.

Програмний продукт відповідає функціональним вимогам, описаним у технічному завданні до дипломного проекту.

ПЕРЕЛІК ПОСИЛАНЬ

1. Опис бібліотеки jQuery: jQuery API [Електронний ресурс] // jquery.com. – 2020. – Режим доступу до ресурсу: https://api.jquery.com/
2. Опис бібліотеки Backbone.js: Backbone.js [Електронний ресурс] // backbonejs.org. – 2020. – Режим доступу до ресурсу: https://backbonejs.org/
3. Опис бібліотеки D3.js: Data-Driven Documents [Електронний ресурс] // d3js.org – 2020. – Режим доступу до ресурсу: https://d3js.org/
4. Опис бібліотеки React: React documentation and related resources. [Електронний ресурс] // reactjs.org– 2020. – Режим доступу до ресурсу: https://reactjs.org/docs/getting-started.html
5. Опис бібліотеки Underscore: underscore documentation. [Електронний ресурс] // underscorejs.org– 2020. – Режим доступу до ресурсу: https://underscorejs.org/
6. Опис бібліотеки MomentJS: MomentJs documentation. [Електронний ресурс] // momentjs.com – 2020. – Режим доступу до ресурсу: https://momentjs.com/
7. Опис бібліотеки Lodash: Lodash documentation. [Електронний ресурс] // lodash.com – 2020. – Режим доступу до ресурсу: https://lodash.com/docs/4.17.15
8. Опис бібліотеки Bootstrap: Build fast, responsive sites with Bootstrap. [Електронний ресурс] // getbootstrap.com – 2020. – Режим доступу до ресурсу: https://getbootstrap.com/docs/4.5/getting-started/introduction/
9. Опис бібліотеки Foundation: The most advanced responsive front-end framework in the world.. [Електронний ресурс] // get.foundation – 2020. – Режим доступу до ресурсу: https://get.foundation/frameworks-docs.html

10. Опис бібліотеки Semantic ui: UI Docs.. [Електронний ресурс] // semantic-ui.com – 2020. – Режим доступу до ресурсу: https://semantic-ui.com/introduction/getting-started.html
11. Опис бібліотеки UI Kit: overview of UIkit. [Електронний ресурс] // getuikit.com – 2020. – Режим доступу до ресурсу: https://getuikit.com/docs/introduction
12. Опис мови Ruby: Guides, tutorials, and reference material to help you learn more about Ruby. [Електронний ресурс] // ruby-lang.org/ – 2020. – Режим доступу до ресурсу: https://www.ruby-lang.org/en/documentation/
13. Опис фреймворку Angular: Introduction to the Angular Docs. [Електронний ресурс] // angular.io / – 2020. – Режим доступу до ресурсу: https://angular.io/docs
14. Опис фреймворку Ember.js: A framework for ambitious web developers. [Електронний ресурс] // emberjs.com / – 2020. – Режим доступу до ресурсу: https://emberjs.com/Express.js
15. Опис фреймворку Express.js: Fast, unopinionated, minimalist web framework for Node.js [Електронний ресурс] // expressjs.com/ – 2020. – Режим доступу до ресурсу: https://expressjs.com/
16. Документація для середовища Node js: API Reference Documentation/. [Електронний ресурс] // nodejs.org / – 2020. – Режим доступу до ресурсу: https://nodejs.org/en/docs/
17. Опис документації javascript компілятора Babel: What is Babel [Електронний ресурс] // babeljs.io / – 2020 – Режим доступу до ресурсу: https://babeljs.io/docs/en/
18. Опис роботи клієнт-серверної архітектури: An introduction to web applications architecture: An introduction to web applications architecture [Електронний ресурс] // www.open.edu / 2020 – Режим доступу до ресурсу: https://www.open.edu/openlearn/science-maths-

technology/introduction-web-applications-architecture/content-section-1.1

19. Опис роботи MVC патерну програмування: Честный MVC на React + Redux [Електронний ресурс] // habr.com / 2020 – Режим доступу до ресурсу: https://habr.com/ru/company/devexpress/blog/305812/
20. Опис роботи REST API технології: Learn REST: A RESTful Tutorial [Електронний ресурс] //restapitutorial.com/ 2020 – Режим доступу до ресурсу: https://www.restapitutorial.com/
21. Опис методів для запиту на сервер: HTTP request methods [Електронний ресурс] // developer.mozilla.org/ 2020 – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods
22. Опис того, що представляють собою класи у javascript: Класс: базовый синтаксис [Електронний ресурс] // learn.javascript.ru / 2020 – Режим доступу до ресурсу: https://learn.javascript.ru/class#ne-prosto-sintaksicheskiy-sahar
23. Опис призначення діаграми послідовності: What is Sequence Diagram? [Електронний ресурс] // visual-paradigm.com / 2020 – Режим доступу до ресурсу: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/

Додаток А

Тексти програмного коду

Клієнт-серверна Web-система для роботи з фінансами

(Найменування програми (документа))

DVD-R

(Вид носія даних)

14 арк, 53 Кб

(Обсяг програми (документа) , арк.,) Кб)

Київ – 2020 року

App.js

```
import React, { Component } from 'react';
import './App.sass';
import Box from '../Box/Box'
import Auth from '../Auth/Auth'
import CommingSoon from '../CommingSoon/CommingSoon'
import Home from '../Home/Home'
import Registration from '../Registration/Registration'
import ConfirmEmail from '../ConfirmEmail/ConfirmEmail'
import { Route, Switch, BrowserRouter as Router } from 'react-router-
dom'

class App extends Component {
  render() {
    return (
      <div className="App">
        <Box>
          <Router>
            <Switch>
              <Route path="/auth" component={Auth} />
              <Route path="/registration" component={Registration} />
              <Route path="/confirm-email" component={ConfirmEmail} />
              <Route path="/comming-soon" component={CommingSoon} />
              <Route path="/" component={Home} />
            </Switch>
          </Router>
        </Box>
      </div>
    );
  }
}
export default App;
```

App.sass

```
.App
  background: linear-gradient(125deg, #374049, #232323)
  width: 100%
  height: 100vh
  display: flex
  justify-content: center
  align-items: center
  padding: 100px
  box-sizing: border-box
```

Home.js

```
import React from 'react'
import './Home.sass'
import Logo from '../../img/Logo_white.svg'
import Background from '../../img/Personal_Cabinet 4.png'

export default function (props) {
  return (
    <>
      <header className="header">
        <div className="logo">
          <img className="logo__img" alt='' src={Logo} />
          <div className="logo__text">save your money</div>
        </div>
        <ul className="info-nav">
          <li className="info-nav__item">Contact us</li>
          <li className="info-nav__item">What we offer</li>
          <li className="info-nav__item">Pricing</li>
        </ul>
      </header>
      <div className="dynamic-content">
        <div className="information">
          <p className="information__item">This is a breakthrough
service to control your finances.</p>
          <p className="information__item">Now you can accurately
plan, control and manage your budget.</p>
          <p className="information__item">It will protect you from
useless expenses.</p>
```

```
          <p className="information__item">Enjoy it..</p>
        </div>

        <div className="buttons">
          <a href="1" className="buttons__item buttons__log-in">Log
in</a>
          <a href="1" className="buttons__item buttons__sign-in">Sign
in</a>
        </div>

        <img src={Background} alt="" className="bgimg" />
      </div>
    </>
  )
}
```

Home.js

```
import React from 'react'
import './Home.sass'
import Logo from '../../img/Logo_white.svg'
import Background from '../../img/Personal_Cabinet 4.png'

export default function (props) {
  return (
    <>
      <header className="header">
        <div className="logo">
          <img className="logo__img" alt='' src={Logo} />
          <div className="logo__text">save your money</div>
        </div>
        <ul className="info-nav">
          <li className="info-nav__item">Contact us</li>
          <li className="info-nav__item">What we offer</li>
          <li className="info-nav__item">Pricing</li>
        </ul>
      </header>
      <div className="dynamic-content">
        <div className="information">
          <p className="information__item">This is a breakthrough
service to control your finances.</p>
```

```
          <p className="information__item">Now you can accurately
plan, control and manage your budget.</p>
          <p className="information__item">It will protect you from
useless expenses.</p>
          <p className="information__item">Enjoy it..</p>
        </div>

        <div className="buttons">
          <a href="1" className="buttons__item buttons__log-in">Log
in</a>
          <a href="1" className="buttons__item buttons__sign-in">Sign
in</a>
        </div>

        <img src={Background} alt="" className="bgimg" />
      </div>
    </>
  )
}
```

Home.sass

```
.header
  display: flex
  font-size: 1.6rem
  justify-content: space-between

.info-nav
  padding-top: 1rem
  opacity: 10%
  &__item
    margin-left: 5rem
    color: #c6c6c6
    font-weight: 400
    font-family: Monserrat, sans-serif

.info-nav
```

```
  display: flex

.logo
  &__img
    width: 16rem
  &__text
    font-family: Montserrat, sans-serif
    font-size: 1.6rem
    font-weight: bold
    color: #c6c6c6
    font-style: italic
    margin-top: 8px

.dynamic-content
  display: flex
  flex-direction: column
  justify-content: space-between

.information
  width: 40rem
  font-size: 1.2rem
  line-height: 1.6
  font-family: Montserrat, sans-serif
  font-style: italic
  color: #c6c6c6
  font-weight: 500
  &__item:not(:last-child)
    margin-bottom: 2rem

.buttons
  margin-top: 10rem
  font-family: Montserrat, sans-serif
  display: flex
  flex-direction: column
  &__item
    width: 20rem
    text-align: center
    padding: 2rem 0
    border-radius: 6px
```

```
    box-sizing: border-box
    font-size: 1.4rem
    box-shadow: 0 0 10px 0 rgba(0, 0, 0, 0.5)
    color: white
    font-weight: bold
  &__item:not(:last-child)
    margin-bottom: 2rem
  &__log-in
    background: linear-gradient(287deg, #ff8500 0%, #efa500 100%)
  &__sign-in
    background: linear-gradient(286deg, #252b30 0%, #2d3339 100%)

.bgimg
  position: absolute
  left: 50rem
  top: 8rem
  width: 120rem
  transform: rotate(-5deg)
  filter: drop-shadow( 0 0 20px rgba(0, 0, 0, 0.5))
```

reset.sass

```
.header
  display: flex
  font-size: 1.6rem
  justify-content: space-between

.info-nav
  padding-top: 1rem
  opacity: 10%
  &__item
    margin-left: 5rem
    color: #c6c6c6
    font-weight: 400
    font-family: Monserrat, sans-serif

.info-nav
  display: flex

.logo
```

```
    &__img
      width: 16rem
    &__text
      font-family: Montserrat, sans-serif
      font-size: 1.6rem
      font-weight: bold
      color: #c6c6c6
      font-style: italic
      margin-top: 8px

  .dynamic-content
    display: flex
    flex-direction: column
    justify-content: space-between

  .information
    width: 40rem
    font-size: 1.2rem
    line-height: 1.6
    font-family: Montserrat, sans-serif
    font-style: italic
    color: #c6c6c6
    font-weight: 500
    &__item:not(:last-child)
      margin-bottom: 2rem

  .buttons
    margin-top: 10rem
    font-family: Montserrat, sans-serif
    display: flex
    flex-direction: column
    &__item
      width: 20rem
      text-align: center
      padding: 2rem 0
      border-radius: 6px
      box-sizing: border-box
      font-size: 1.4rem
      box-shadow: 0 0 10px 0 rgba(0, 0, 0, 0.5)
```

```
    color: white
    font-weight: bold
  &__item:not(:last-child)
    margin-bottom: 2rem
  &__log-in
    background: linear-gradient(287deg, #ff8500 0%, #efa500 100%)
  &__sign-in
    background: linear-gradient(286deg, #252b30 0%, #2d3339 100%)

.bgimg
  position: absolute
  left: 50rem
  top: 8rem
  width: 120rem
  transform: rotate(-5deg)
  filter: drop-shadow( 0 0 20px rgba(0, 0, 0, 0.5))
```

Box.js

```
import React from 'react'
import './Box.sass'

const Box = props => {
  return (
    <div className='content-box'>
      {props.children}

    </div>
  )
}

export default Box
```

Box.sass

```
.content-box
  padding: 8rem
  width: 1400px
  height: 800px
  border-radius: 80px
  background: linear-gradient(124deg, #161a1f 3%, #2d3339 99%)
```

```
  box-shadow: 0 0 68px 0 rgba(0, 0, 0, 0.5)
  display: flex

  flex-direction: column
  justify-content: space-between
  box-sizing: border-box
  position: relative
  overflow: hidden
```

baseRepository.js

```
const { dbAdapter } = require('../config/db');
const { v4 } = require('uuid');


class BaseRepository {
    constructor(collectionName) {
        this.dbContext = dbAdapter.get(collectionName);
        this.collectionName = collectionName;
    }

    generateId() {
        return v4();
    }

    getAll() {
        return this.dbContext.value();
    }

    getOne(search) {
        return this.dbContext.find(search).value();
    }

    create(data) {
        data.id = this.generateId();
        data.createdAt = new Date();
        const list = this.dbContext.push(data).write();
        return list.find(it => it.id === data.id);
    }
```

```
    update(id, dataToUpdate) {
        dataToUpdate.updatedAt = new Date();
        return this.dbContext.find({ id
}).assign(dataToUpdate).write();
    }

    delete(id) {
        return this.dbContext.remove({ id }).write();
    }
}

exports.BaseRepository = BaseRepository;
```

UserRepository.js

```
const { BaseRepository } = require('./baseRepository');

class UserRepository extends BaseRepository {
    constructor() {
        super('users');
    }
}

exports.UserRepository = new UserRepository();
```

UserValidationmiddlevare.js

```
const { user } = require('../models/user');


const validateEmail = (inputText) => {
    return (/@gmail\.com$/.test(inputText))
}

const validatePhone = (phoneNumber) => {
    if (phoneNumber) {
        let phoneAfterCode = phoneNumber.slice(4)
        let phoneCode = phoneNumber.slice(0, 4)
        let isnum = (/^\d+$/.test(phoneAfterCode) &&
phoneAfterCode.length == 9 && phoneCode == "+380");
        return isnum
```

```
    }
    else {
        return 0
    }
}

const passValidator = (password) => {
    if (password) {
        const isPass = password.length >= 3;
        return isPass
    }
    else return 0

}


const removeExcessFields = (body) => {
    for (let key in body) {
        if (!user.hasOwnProperty(key)) {
            delete body[key]
        }
    }
}

const checkForAllFields = (body) => {
    for (let key in user) {
        if (!body.hasOwnProperty(key) && key !== 'id') {
            return false
        }
    }
    return true
}

const createUserValid = (req, res, next) => {

    removeExcessFields(req.body)
    const allRequiredFields = checkForAllFields(req.body);
    const isGmail = validateEmail(req.body.email);
    const isPhone = validatePhone(req.body.phoneNumber);
    const isPass = passValidator(req.body.password);
```

```
    const idExist = !!req.body.id

    if (!allRequiredFields) {
        res
            .status(400)
            .send({
                error: true,
                message: 'You are miss some fields in you request'
            })
    }

    else if (idExist) {
        res
            .status(400)
            .send({
                error: true,
                message: 'In reques exist ID, please remove it from
body'
            })
    }
    else if (!isGmail || !isPhone || !isPass) {
        let message = `${!isGmail ? "Gmail, " : ""}${!isPhone ?
"Phone, " : ""}${!isPass ? "Pass, " : ""}not valid`
        res
            .status(400)
            .send({
                error: true,
                message: message
            })
    } else {
        next()
    }
}

const updateUserValid = (req, res, next) => {

    removeExcessFields(req.body)
    const allRequiredFields = checkForAllFields(req.body);
    const isValidGmail = validateEmail(req.body.email);
```

```
    const isValidPhone = validatePhone(req.body.phoneNumber);
    const isValidPass = passValidator(req.body.password);
    const idExist = !!req.body.id

    if (!req.body) {
        res
            .status(400)
            .send({
                error: true,
                message: 'You are miss some fields in you request'
            })
    } else if (idExist) {
        res
            .status(400)
            .send({
                error: true,
                message: 'In reques exist ID, please remove it from
body'
            })
    } else if (!isValidGmail || !isValidPhone || !isValidPass) {
        let message = `${!isValidGmail ? "Email, " :
""}${!isValidPhone ? "Phone, " : ""}${!isValidPass ? "Pass, " : ""}not
valid`
        res
            .status(400)
            .send({
                error: true,
                message: message
            })
    }
    else {
        next()
    }
}

exports.createUserValid = createUserValid;
exports.updateUserValid = updateUserValid;
```

user.routes.js

```
const { Router } = require('express');
const UserService = require('../services/userService');
const { createUserValid, updateUserValid } =
require('../middlewares/user.validation.middleware');
const { responseMiddleware } =
require('../middlewares/response.middleware');

const router = Router();

// TODO: Implement route controllers for user

router.get('/', function (req, res, next) {
  res.body = UserService.searchAll()
  res.error = `There are no users on server`
  next()
}, responseMiddleware)


router.get('/:id', function (req, res, next) {
  res.body = UserService.search(req.params.id)
  res.error = `User with id:${req.params.id} not found`
  next()
}, responseMiddleware)


router.post('/', createUserValid, function (req, res, next) {
  res.body = UserService.createUsers(req.body)
  res.error = `User already exist`
  next()
}, responseMiddleware)
router.put('/:id', updateUserValid, function (req, res, next) {
  res.body = UserService.updateUser(req.params.id, req.body)
  res.error = `User is not found`
  next()
}, responseMiddleware)


router.delete('/:id', function (req, res, next) {
  res.body = UserService.deleteUser(req.params.id)
  res.error = `User is not found`
```

```
  next()
}, responseMiddleware)

module.exports = router;
```

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО”
Кафедра автоматизованих систем обробки інформації та управління

УЗГОДЖЕНО
Керівник проєкту
_____ *Оксана ЖУРАКОВСЬКА*
(підпис) (вл. ім’я, прізвище)
“13” квітня 2020 р.

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ *Олександр ПАВЛОВ*
(підпис) (вл. ім’я, прізвище)
“14” квітня 2020 р.

КЛІЄНТ-СЕРВЕРНА WEB-СИСТЕМА ДЛЯ РОБОТИ З ФІНАНСАМИ

ТЕХНІЧНЕ ЗАВДАННЯ

Шифр *ДП 6111.01.000 ТЗ*

на 10 сторінках

Київ – 2020 року

ЗМІСТ

1 ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1 Повне найменування системи та її умовне позначення

Повне найменування системи: «Клієнт-серверна Web-система для роботи з фінансами»

Коротке найменування системи: «Клієнт-серверна Web-система для роботи з фінансами»

1.2 Найменування організації замовника та організації учасників робіт

Замовником системи є кафедра автоматизованих систем обробки інформації та управління факультету інформатики та обчислювальної техніки Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського.» Представник замовника: доцент кафедри. Жураковська Оксана Сергіївна. Адрес замовника: м. Київ, п-кт Перемоги 37.

Розробником системи є студент групи ІС-61 кафедри автоматизованих систем обробки інформації та управління Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» Касьян Євгеній Володимирович.

1.3 Перелік документів, на підставі яких створюється система

При розробці системи і створення проектно-експлуатаційної документації виконавець повинен керуватися вимогами наступних нормативних документів:

- ДСТУ 19.201-78. Технічне завдання. Вимоги до змісту і оформлення;
- ДСТУ 34.601-90. Комплекс стандартів на автоматизовані системи. Автоматизовані системи. Стадії створення;

- ДСТУ 34.201-89. Інформаційні технології. Комплекс стандартів на автоматизовані системи. Види, комплексність і позначення документів при створенні автоматизованих систем.

1.4 Планові терміни початку і закінчення роботи зі створення системи

Плановий строк початку роботи по створенню системи 7 березня 2020 року. Плановий строк кінця роботи по створенню системи для контролю особистих фінансів 31 травня 2020 року.

2 ПРИЗНАЧЕННЯ І ЦІЛІ СТВОРЕННЯ СИСТЕМИ

2.1 Призначення системи

Система спрямована на спрощення ведення обліку витрат і доходів людини. Це дозволить отримати чітке уявлення про рух особистих грошей. Людина повинна розуміти, на що вона витрачається, які бюджетні статті залишаються стабільними, а які постійно змінюються. Управління особистими фінансами неможливо без чіткої картини. Щоб правильно провести процес моніторингу, можна скористатися, розроблюємою системою.

2.2 Мета та задачі створення системи

Метою розробки системи є спрощення процесу обліку особистого грошового потоку та фінансового контролю.

Для досягнення поставленої мети потрібно вирішити такі задачі:

- реєстрація та авторизація користувачів;
- створення журналу фінансових операцій;
- планування бюджету на обраний період;
- додавання категорій за якими будуть здійснюватися операції.

3 ХАРАКТЕРИСТИКА ОБ'ЄКТА АВТОМАТИЗАЦІЇ

Об'єктом є процес, спрямований на автоматизацію ведення обліку фінансів. Автоматизована система має допомагати простіше вести облік та економити час на обчислення всіх математичних операцій які постають у задачі ведення фінансів, та спростити управління своїми коштами.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

Функціональні вимоги до системи:

- система надає можливість користувачу зареєструватися у системі;
- система надає можливість користувачу увійти у особистий кабінет;
- система надає можливість користувачу додати транзакцію для облікового рахунку;
- система надає можливість користувачу переглянути інформацію що до використання системи;
- система надає можливість користувачу зберегти введенні данні до бази даних;
- система надає можливість сформувати бюджет на обраний період;

4.2 Вимоги до надійності

Користувач, який працює з програмою через веб-браузер, повинен мати постійний доступ до веб-програми, розташованої за конкретним URL.

Система повинна зберігати приватну інформацію користувача у зашифрованому вигляді та забезпечити цілісність інформації, та захист від несанкціонованого доступу

4.3 Умови експлуатації

Для використання програмного забезпечення, користувач має мати стабільне підключення до інтернету та встановлений браузер. Доступ до сервісу надається через Web – застосунок.

Також всі данні які вводить користувач повинні валідуватись на правильність

4.4 Вимоги до складу і параметрів технічних засобів

Для коректного функціонування системи було виділено наступні вимого до технічного забезпечення:

1. Мінімальні технічні вимоги до компьютеру:
   - Мінімальна кількість оперативної пам’яті 4gb;
   - Процесор з тактовою частотою не нижче 1Ггц;
   - Пристрої для вводу та виводу інфорації (мишка або трекпад, монітор, клавіатура).
2. Вимоги до встановленого програмного забезпечення:
   - База даних mongoDB;
   - Web-framework Exprress;
   - NodeJS;
   - React, React-DOM.

5 СТАДІЇ ТА ЕТАПИ РОЗРОБКИ

У таблиці 5.1 наведено календарний план робіт та терміни їх виконання.

Таблиця 5.1 – Календарний план виконання робіт

№ з/п	Назва етапів розробки програмного продукту	Кінцевий термін виконання
1.	Визначення теми дипломного проекту та погодження з дипломним керівником	01.03.2020
2.	Опис предметного середовища, процесу діяльності, функціональної моделі	8.03.2020
3.	Моделювання структурної схеми моделі використань	8.03.2020
4.	Огляд наявних аналогів	15.03.2020
5.	Опис постановки задачі. Визначення задач та цілей розробки	15.03.2020
6.	Розроблення дизайну програмного застосування	22.03.2020
7.	Опис інформаційного забезпечення (опис вхідних даних, вихідних даних, Опис структури бази даних	22.03.2020
8.	Розробка структурної схеми бази даних	29.03.2020
9.	Розробка візуального інтерфейсу	5.04.2020
10.	Розробка Front-end частини програмного забезпечення	12.04.2020
11.	Розробка Back-end частини програмного забезпечення	19.04.2020
12.	Додавання бази даних до ПЗ	26.04.2020
13.	Опис математичного забезпечення	3.05.2020
14.	Опис програмного та технічного забезпечення	10.05.2020
15.	Тестування системи, виявлення та усунення недоліків	17.05.2020

6 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ СИСТЕМИ

6.1 Види випробувань

Види випробувань узгоджуються з замовником до проведення випробувань.

Здача програмного забезпечення відбувається на комп’ютері виконавця завдання з встановленим програмним забезпеченням, що відповідає вимогам.

Всі випробування, які проводилися для перевірки програмного продукту, наведені у пояснювальній записці до дипломного проекту. Методика тестувань наведена детально у пояснювальній записці. Випробуванням підлягає перевірка основних функцій системи. Було проведене функціональне, модульне та інтеграційне тестування.

UNICHECK

Власник документу:
Попенко Володимир Дмитрович

Дата перевірки:
08.06.2020 18:25:13 EEST

Дата звіту:
09.06.2020 17:07:53 EEST

ID перевірки:
1003882091

Тип перевірки:
Doc vs Internet + Library

ID користувача:
77149

Назва документу: Kasyan_bachelor_is61

ID файлу: 1003896992 **Кількість сторінок:** 54 **Кількість слів:** 6984 **Кількість символів:** 52709 **Розмір файлу:** 26.31 MB

6.14% Схожість

Найбільша схожість: 1.68% з джерело бібліотеки. **ID файлу:** 1000044113

2.23% Схожість з Інтернет джерелами	17	Page 56
5.67% Текстові збіги по Бібліотеці акаунту	127	Page 56

0% Цитат

Не знайдено жодних цитат

0% Вилучень

Вилучений текст відсутній

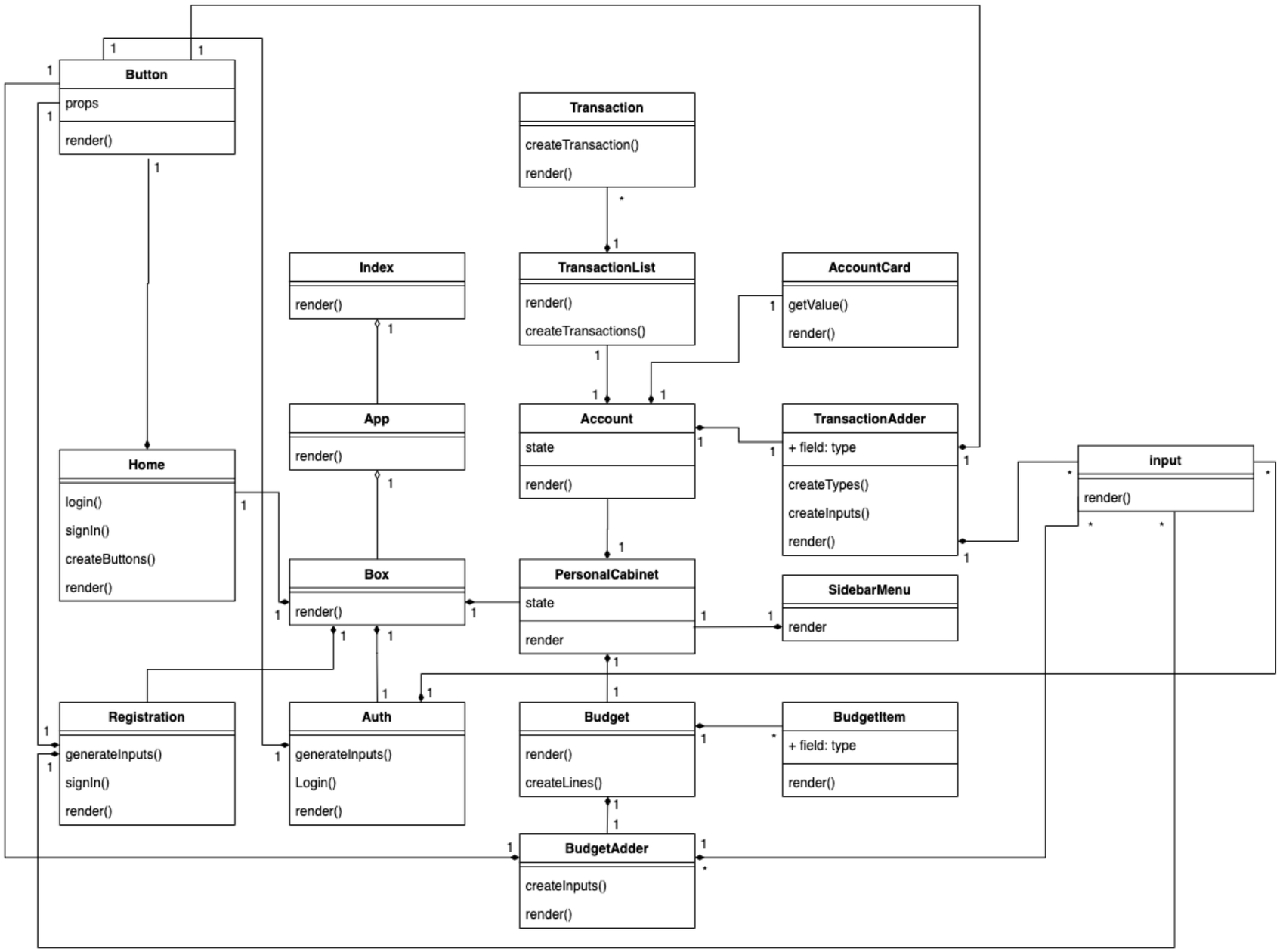
Підміна символів

Не знайдено замінених символів

Графічний матеріал
до дипломного проєкту

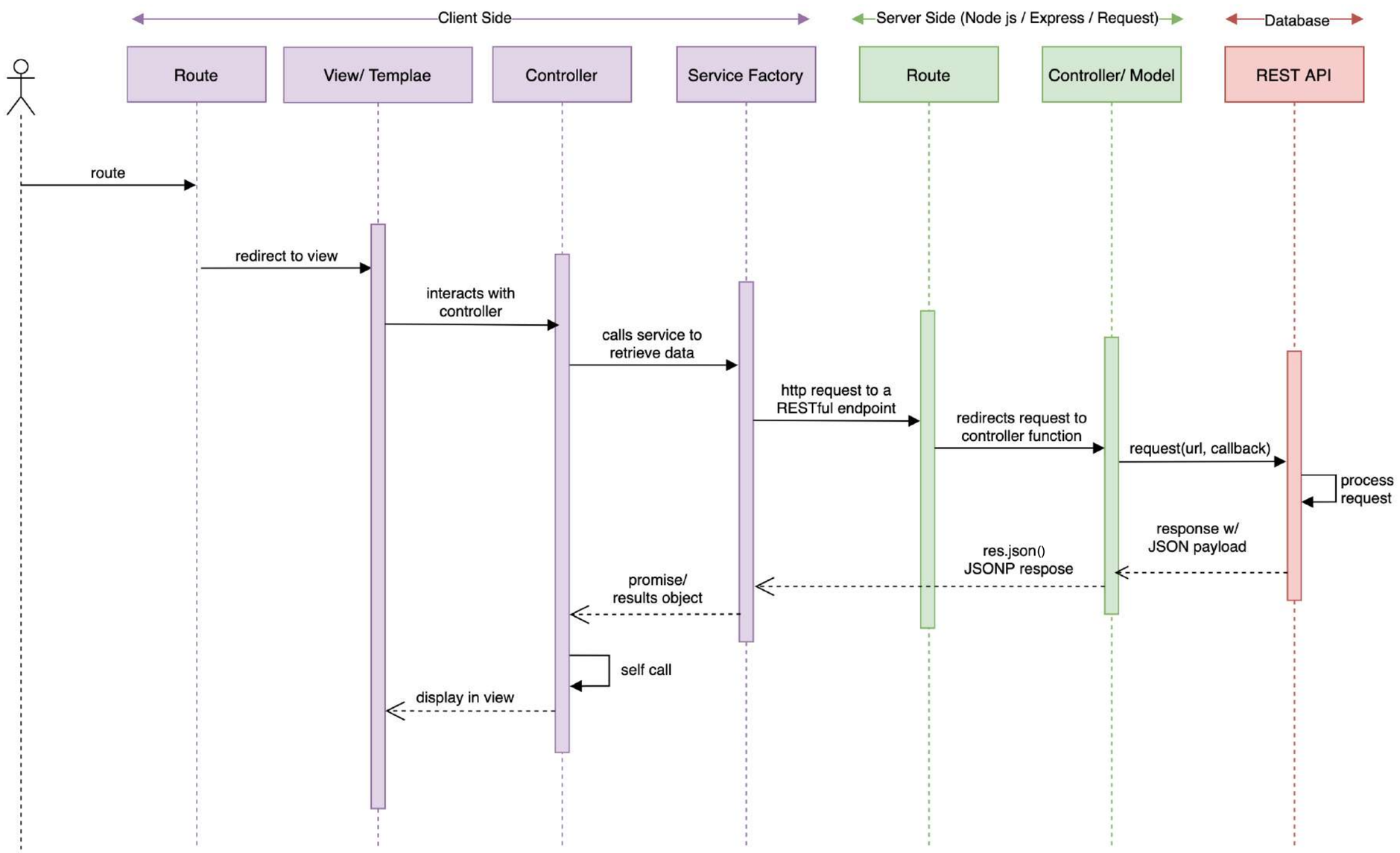
на тему: Клієнт-серверна Web-система для роботи з фінансами

Київ – 2020 року



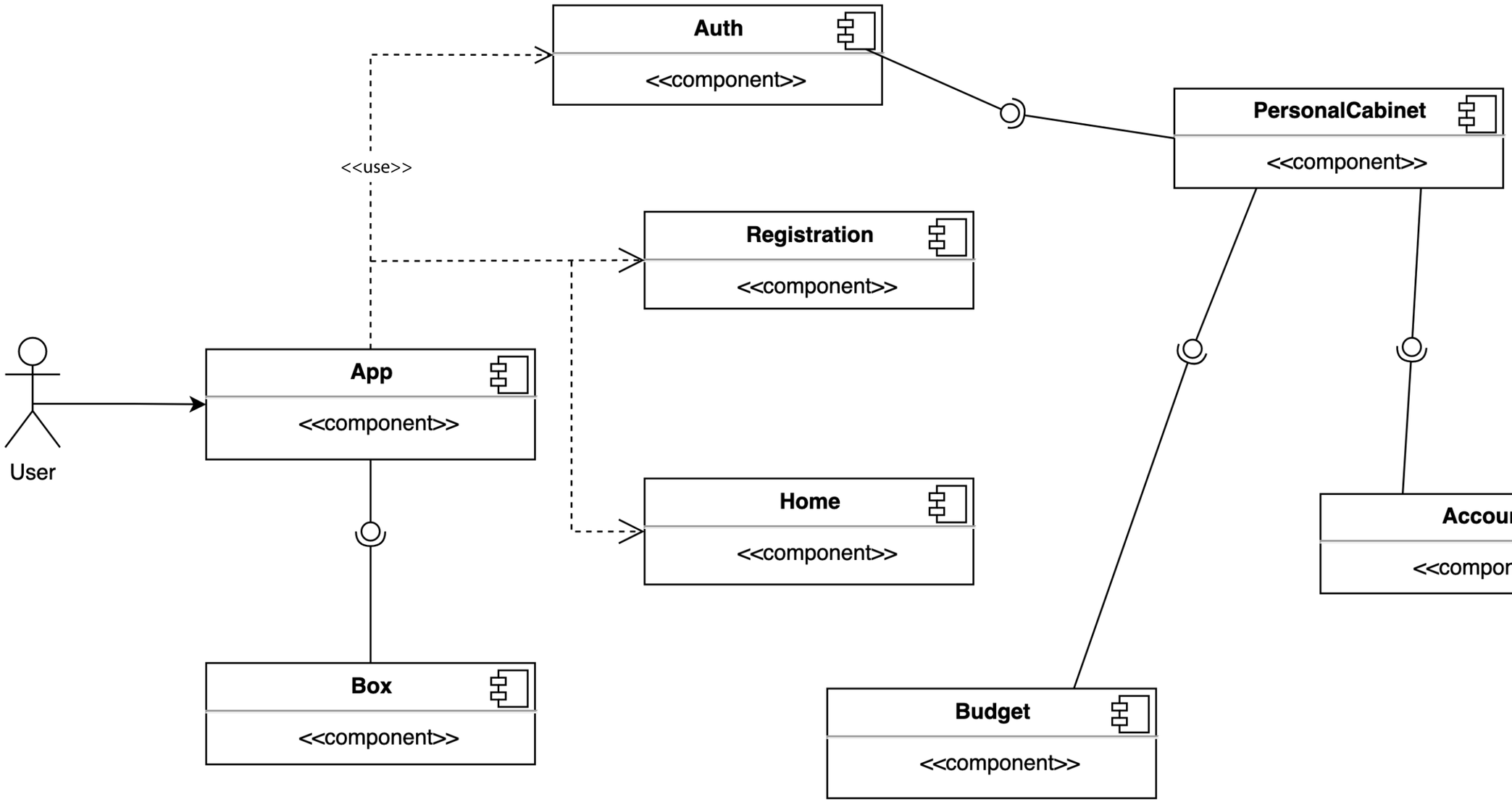
					ДП 6111.02.000 ССК	
Зм.	Арк.	№ документа	Підпис	Дата	*Схема структурна класів програмного забезпечення*	Літера / Маса / Масштаб
Розробив		Касьян Є. В.				
Перевірив		Жураковська О.С.				Аркуш 1 / Аркушів 1
Т. кон.						
Н. кон.		Тєлишева Т.О.			*Клієнт-серверна Web-система для роботи з фінансами*	*КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-61*
Затвердив		Жураковська О.С.				

ДП 6111.03.000 ССП



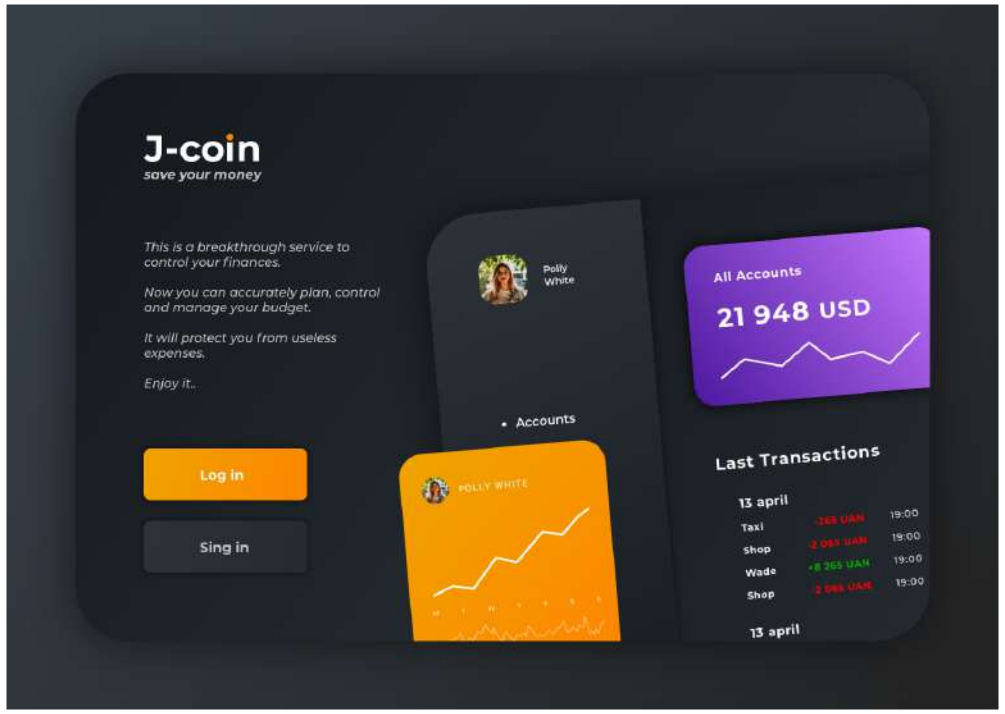
					ДП 6111.03.000 ССП		
					Схема структурна послідовності	Літера	Маса / Масштаб
Зм.	Арк.	№ документа	Підпис	Дата			
Розробив		Касьян Є. В.				Аркуш 1	Аркушів 1
Перевірив		Жураковська О.С.					
Т. кон.							
					Клієнт-серверна Web-система для роботи з фінансами	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-61	
Н. кон.		Тєлишева Т.О.					
Затвердив		Жураковська О.С.					

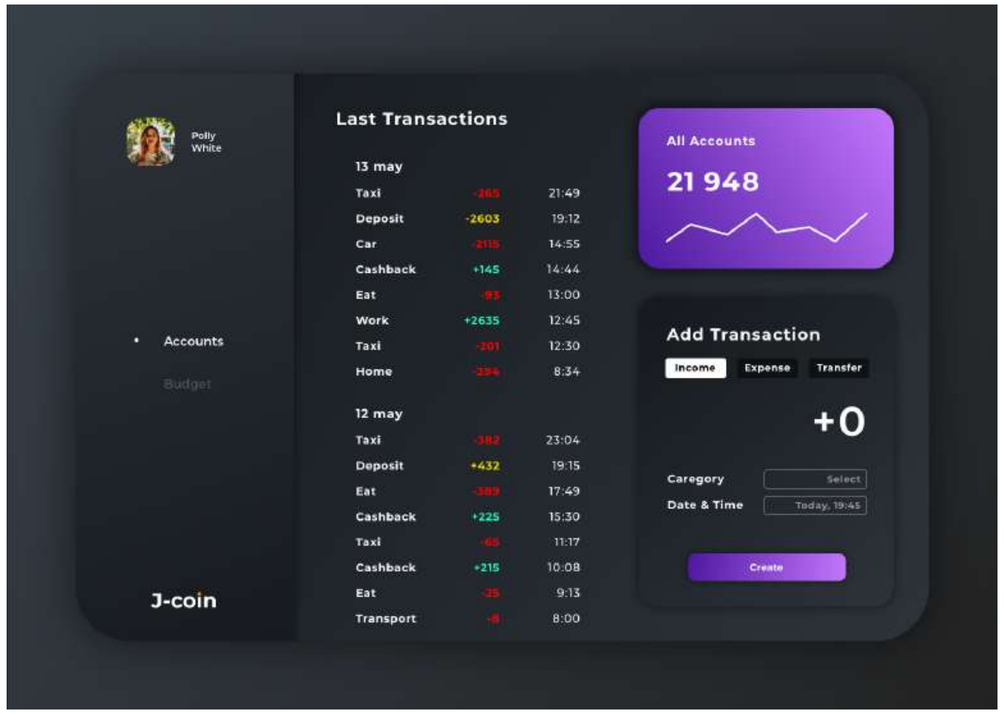
ДП 6111.04.000 ССК

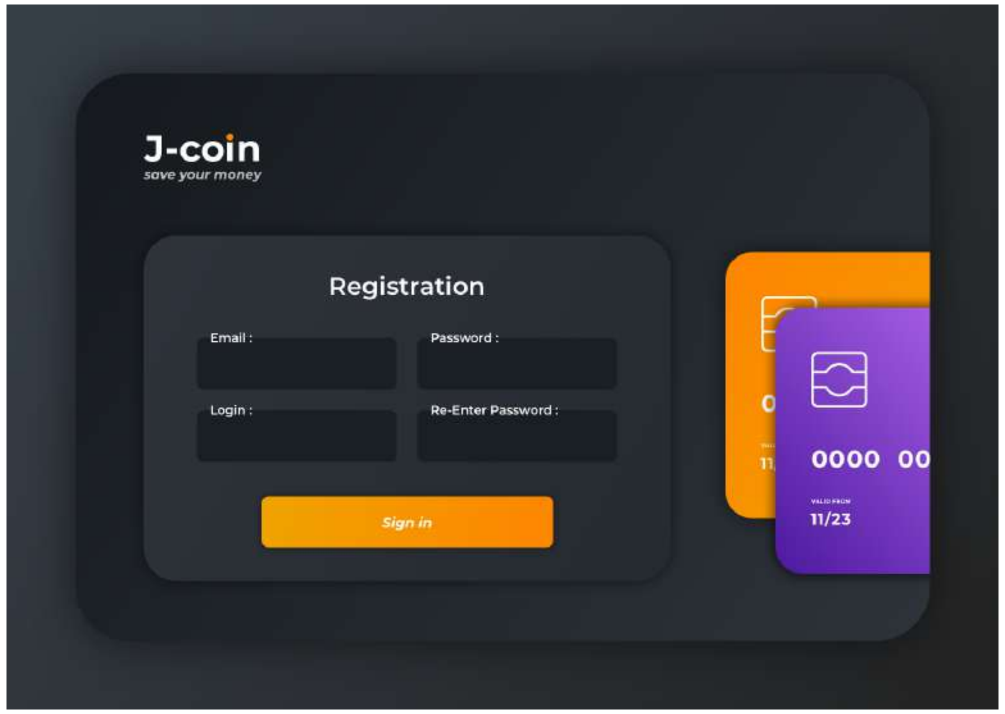


					ДП 6111.04.000 ССК		
						Літера	Маса	Масштаб
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна компонентів програмного забезпечення			
Розробив		Касьян Є. В.				Аркуш 1	Аркушів 1
Перевірив		Жураковська О.С.					
Т. кон.							
					Клієнт-серверна Web-система для роботи з фінансами	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-61	
Н. кон.		Тєлишева Т.О.					
Затвердив		Жураковська О.С.					

ДП 6111.05.000 КЕ

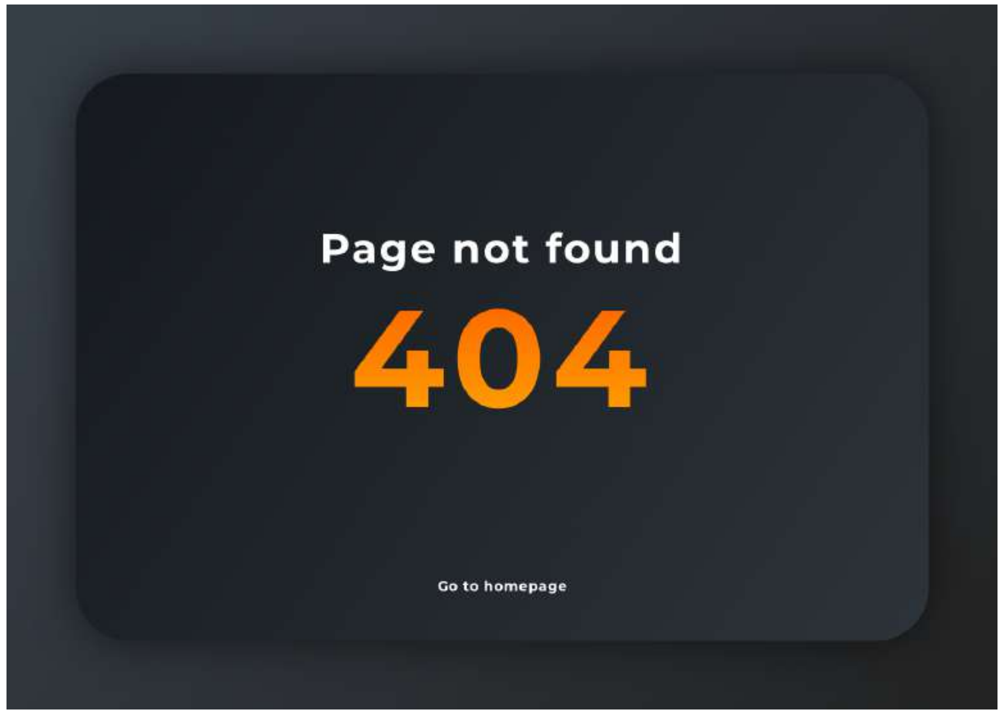


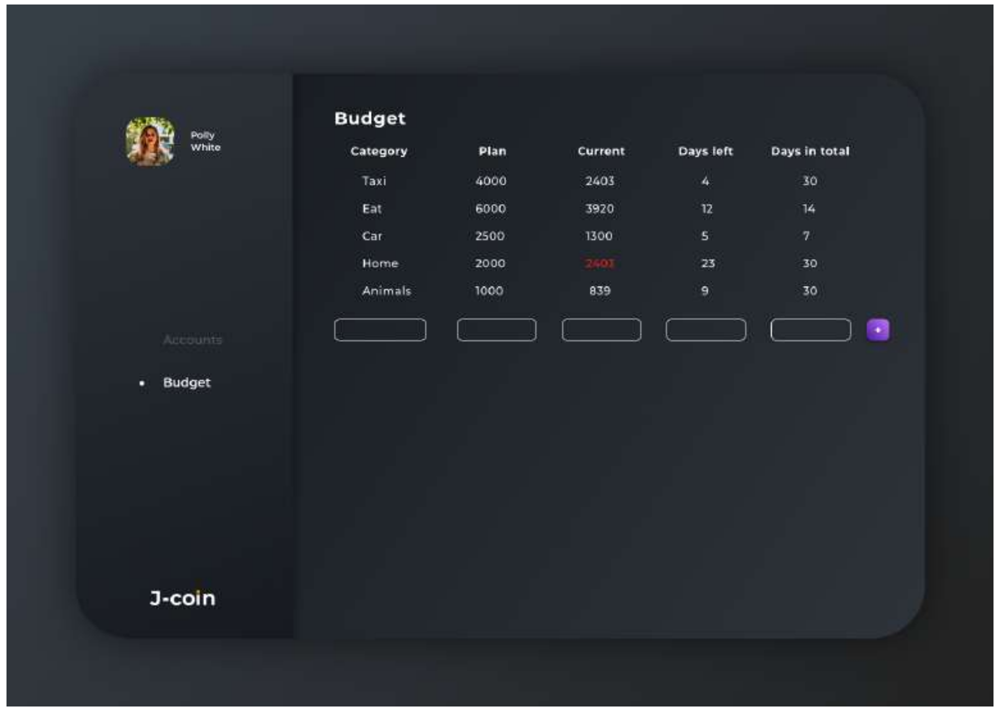




					ДП 6111.05.000 КЕ		
Зм.	Арк.	№ документа	Підпис	Дата	*Креслення вигляду екранних форм*	Літера	Маса	Масштаб
Розробив		Касьян Є. В.						
Перевірив		Жураковська О.С.				Аркуш 1	Аркушів 2	
Т. кон.								
Н. кон.		Тєлишева Т.О.			*Клієнт-серверна Web-система для роботи з фінансами*	*КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-61*		
Затвердив		Жураковська О.С.						

ДП 6111.05.000 КЕ





					ДП 6111.05.000 КЕ		
Зм.	Арк.	№ документа	Підпис	Дата	Креслення вигляду екранних форм	Літера	Маса	Масштаб
Розробив		Касьян Є. В.					
Перевірив		Жураковська О.С.				Аркуш 2	Аркушів 2
Т. кон.					Клієнт-серверна Web-система для роботи з фінансами	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-61	
Н. кон.		Тєлишева Т.О.					
Затвердив		Жураковська О.С.					